공학박사 학위논문

영상 super-resolution을 위한 CNN 하드웨어의 on-chip 메모리 최적화

On-chip memory reduction in CNN hardware design for image super-resolution

2019 년 2 월

서울대학교 대학원

전기 컴퓨터 공학부

이 동 현

영상 super-resolution을 위한 CNN 하드웨어의 on-chip 메모리 최적화

On-chip memory reduction in CNN hardware design for image super-resolution

지도 교수 이 혁 재

이 논문을 공학박사 학위논문으로 제출함

2019 년 2월

서울대학교 대학원

전기 컴퓨터 공학부

이 동 현

이동현의 공학박사 학위논문을 인준함

2019 년 2 월

위 원 장 최 기 영 (인)

부위원장 이 혁 재 (인)

위 원 조 남 익 (인)

위 원 권 영 준 (인)

위 원 이 규 중 (인)

영상 super-resolution을 위한 CNN 하드웨어의 on-chip 메모리 최적화

On-chip memory reduction in CNN hardware design for image super-resolution

지도 교수 이 혁 재

이 논문을 공학박사 학위논문으로 제출함

2019 년 2월

서울대학교 대학원

전기 컴퓨터 공학부

이 동 현

이동현의 공학박사 학위논문을 인준함

2019 년 2 월

위 원 장 최 기 영 (인)

부위원장 이 혁 재 (인)

위 원 조 남 익 (인)

위 원 권 영 준 (인)

위 원 이 규 중 (인)

초 록

Single image super-resolution (SISR) 을 위한 convolutional neural network (CNN) 는 영상 분류용 CNN과 달리 고해상도의 영상을 입력 받아 고해상도의 중간 연산 결과인 feature map을 생성한다. SISR용 CNN을 가속하기 위한 하드웨어는 주로 디스플레이 장치에 적용이 되며 외부 메모리 접근이 불가능한 스트리밍 구조를 갖는다. 이는 on-chip 메모리의 용량이 제한적인 하드웨어의 특성상 구현의 어려움을 야기한다. 기존의 연구들은 on-chip 메모리를 감소하기 위해 성능 저하 또는 압축 모듈을 추가한다. 본 논문은 성능 저하 없이 SISR용 CNN 하드웨어의 on-chip 메모리 감소 및 하드웨어를 설계하기 위한 방법을 제안한다.

CNN 하드웨어는 VDSR (Very deep neural network for super-resolution) 구조를 기반으로 한다. 기존 CNN 하드웨어의 SRAM에 읽기 및 쓰기 접근이 동시에 발생하는 래스터 스캔 순서를 부분적 수직 순서로 변경 함으로 읽기 및 쓰기 접근 타이밍을 분리한다. 부분적 수직 순서는 기존의 CNN 하드웨어가 사용하는 듀얼 포트 SRAM 대신 싱글 포트 SRAM을 사용하도록 하며 이는 on-chip 메모리를 절반으로 감소한다. 두 번째 방법으로 VDSR의 필터의 형태를 변경하는 방법을 적용한다. On-chip 메모리의 크기는 컨볼루션 필터의 높이에 비례한다. 그러나 VDSR의 필터는 대칭 구조 중 가장 작은 필터 모양이므로 해당 문제를 해결하기 위해 컨텍스트 보존 1D 필터 구성 방법 및 컨텍스트를 기반한 세로 필터 감소 방법을 적용하여 SRAM의 크기를 절반으로 추가적으로 감소한다.

CNN 하드웨어 구조가 확정 된 이후 CNN의 SISR 성능을 개선

하기 위한 CNN학습 방법을 자연 영상 (natural image)와 텍스트 영상 (text image)에 대해 각각 제안한다. SRGAN (Super-resolution generative adversarial networks) 는 판별자 네트워크 (discriminator network)로부터 발생하는 손실으로 SISR용 CNN이 실제 영상처럼 보이는 자연 영상을 출력하도록 한다. 그러나 SRGAN은 과선명화로 인한 시각적 결함을 발생하는 문제가 있다. 본 논문은 SRGAN의 시각적 결함을 제거하는 두 가지 방법을 제안한다. 첫 번째는 판별자 네트워크의 구조를 변경하여 판별자 네트워크 내에서 영상의 세부 정보 손실을 방지하는 해상도 유지 판별자 네트워크 구조를 제안 한다. 두 번째는 콘텐트 손실을 발생하는 VGG 네트워크의 구조상 영상의 세부적인 정보를 손실하는 문제를 해결하기 위한 해상도 유지 콘텐트 손실 방법을 제안한다.

텍스트 영상은 자연 영상이 아닌 합성 영상으로 영상 내 폰트와 배경의 색상 조합을 다양하게 변경될 수 있다. 기존의 CNN 학습 방법은 네트워크의 일반화를 위해 다양한 종류의 영상을 학습 시키는 방법을 사용한다. 그러나 모든 종류의 색상 조합을 CNN에 학습 시키는 것은 불가능하다. 본 논문은 영상 압축에 사용되는 De-colorization 방법을 차용하여 CNN이 학습할 영상을 검은 폰트와 흰색 배경으로 이루어진 영상으로 한정 함으로 학습되지 않은 영상의 폰트 및 배경 색상 조합에도 시각적 결함 없이 SISR 연산을 수행 하는 방법을 제안 한다.

주요어 : Single image super-resolution, on-chip memory, real-time hardware, generative adversarial networks, text super-resolution

학 번 : 2012-20828

목 차

표 목차

그림 목차

제 1 장 서 론

제 1 절 연구의 배경

단일 영상 초해상도 (Single image super-resolution; SISR)는 낮은 해상도의 영상 (low resolution image; I^{LR}) 을 입력 받아 높은 해상도의 초해상도 영상 (super-resolution image; I^{SR}) 을 출력하는 연산으로 의료 영상, 영상 감시 시스템 (video surveillance system)과 고 해상도 디스플레이 장치 등에 활용 된다 [1-4]. 최근 디스플레이 장치들은 full-HD 해상도를 넘어서 4K ultra-HD와 8K ultra-HD 해상도의 영상을 출력하는 추세이다. 디스플레이 장치들의 출력할 수 있는 최대 해상도는 증가하나 실제 영상 데이터는 디스플레이 장치의 출력 가능한 최대 해상도보다 낮은 경우가 있으며, 이러한 경우 I^{LR} 으로부터 시각적 결함 (visual artifact) 없으며 시각적으로 뛰어난 I^{SR}을 생성 하는 것이 중요하다.

기존의 SISR 방법으로 가장 단순한 방법은 Bicubic 보간법이 있다. Bicubic 보간법은 영상의 단순한 영역에 대해서는 높은 성능을 보이나, 반복되는 패턴이나 텍스트와 같이 복잡한 영역은 blur 된 영상을 출력하는 문제가 있다. SISR에 관한 연구는 영상의 복잡한 영역의 정보를 유지하면서 시각적으로 뛰어난 I^{SR} 을 출력하는 방향으로 진행되며 기계 학습 (machine learning)을 기반으로 한 연구들이 존재한다 [5-8]. 기계 학습 기반의 SISR 은 기존의 비 기계학습 기반의 방법과 비교하여 높은 SISR 성능을 내지만 여전히 영상의 복잡한 영역에 대해서는 시각적 결함을 생성하는 문제가 있다. C. Dong *et al.* 은 영상 분류 (image classification) 분야에서 높은 성능을 보이는

convolutional neural network (CNN)을 이용한 SISR 방법인 super-resolution CNN (SRCNN)을 제안한다 [9]. SRCNN은 기존의 기계 학습 기반의 SISR 들의 성능을 뛰어넘으며 CNN을 기반으로 한 SISR 연구가 주목을 받고 있다.

CNN을 기반으로 한 SISR 방법 중 very deep CNN for super-resolution (VDSR)은 영상 분류 네트워크 중 VGG network [10] 를 기반으로 하여 모든 컨볼루션 레이어의 필터 크기를 3×3으로 구성하여 CNN기반의 SISR 방법 중 높은 성능을 보인다 [11]. VDSR은 20개의 컨볼루션 레이어로 구성되며 각각의 컨볼루션 레이어는 64개의 필터를 갖는 구조이다. VDSR은 영상 분류 딥러닝 (deep learning) 분야에서 적용되는 내용인 CNN의 컨볼루션 레이어 수가 증가 할 수록 성능도 증가함을 보임으로 이후의 SISR 용 CNN 구조들이 해당 내용을 기반으로 더 많은 컨볼루션 레이어로 CNN을 구성하도록 한다. FPGA나 ASIC을 기반으로 한 CNN 가속기에 대한 기존 연구는 영상 분류용 CNN을 가속하기 위한 구조가 주로 연구 된다 [12, 13]. 영상 분류용 CNN은 저해상도 feature map ($fmap$)을 기반으로 연산이 수행되므로 on-chip 메모리의 제약이 적은 반면 SISR용 CNN은 고해상도 $fmap$을 기반으로 연산이 수행되어 많은 양의 on-chip 메모리를 필요로 한다. 표 1.1은 VGG19 network와 VDSR을 비교를 보인다. VGG19는 ImageNet의 224×224 해상도 영상을 사용 하며, VDSR은 SISR 영상의 출력 해상도가 full-HD임을 가정한다. VDSR은 VGG19과 같이 3×3

표 1.1 VGG19와 VDSR 네트워크 비교

	VGG19	VDSR	VDSR/VGG19
Convolution filter	3×3	3×3	
Convolution layers	16	20	
Parameters ($\times 10^3$)	200218.88	664.70	3.32%
Feature maps (word)	323.31K (224×224)	4563.75K (1920×1080)	1411.56%

필터로 구성되며 VGG19와 비교하여 4개의 컨볼루션 레이어를 더 사용한다. 그러나, VDSR의 hidden 컨볼루션 레이어는 모두 64 채널의 $fmap$을 출력하며 이는 VGG19 대비 3.32%의 parameters를 사용한다. 반면에 고해상도의 $fmap$ 을 기반으로 컨볼루션 연산을 수행하므로 VGG19와 비교하여 14.12배 많은 on-chip 메모리를 요구하는 구조이다. 디스플레이 장치는 실시간 동작이 필수로 SISR 연산을 수행하는 스트리밍 구조의 하드웨어를 내장 한다. 따라서 기존의 영상 분류용 CNN의 가속기 구조를 적용하는 것이 불가능하다. 그러나 SISR 하드웨어의 크기는 제약이 있으며 특히 컨볼루션 레이어의 중간 연산 결과 값을 on-chip 메모리에 저장해야 한다. SISR 용 CNN은 고해상도의 $fmap$ 데이터를 on-chip 메모리에 저장하므로 계속하여 증가하는 CNN 의 크기는 SISR용 CNN을 디스플레이 장치에 적용하기에 어렵게 한다.

SISR용 CNN 하드웨어가 고정된 경우 SISR 성능을 높이기 위한 방법은 효과적인 CNN의 학습이다. SISR의 목적은 영상의 세밀한 정보들을 유지하는 I^{SR} 을 생성하는 것이다. 대부분의 연구들이 CNN을 학습하기 위해 고해상도 원본 영상 (high resolution image; I^{HR}) 과 I^{SR} 간 픽셀 간의 차이를 손실 함수로 사용하며, 이는 I^{SR} 과 I^{HR} 간의 평균 픽셀 오차는 감소하지만 I^{SR} 의 세밀한 정보들을 손실하는 문제가 있다. Super-resolution generative adversarial network (SRGAN) [14]은 I^{SR} 의 세밀한 정보 손실을 막기 위해 적대적 생성 신경망 (generative adversarial networks; GANs) [15] 구조를 사용한다. SRGAN은 I^{SR} 을 출력하는 생성자 신경망 (generator network; G)과 I^{SR} 과 I^{HR} 을 구분하는 판별자 신경망 (discriminator network; D)로 구성된다. D는 I^{HR} 과 I^{SR} 을 구분한 후 손실을 G에게 전달함으로 G가

D의 관점에서 원본과 동일한 확률 분포를 갖는 영상을 출력하도록 학습시킨다. 또한, I^{SR} 과 I^{HR} 의 유사도를 증가하기 위해 컨텐트 손실 (content loss) 를 함께 사용한다. SRGAN은 기존의 CNN들과 비교하여 원본과 유사해 보이는 I^{SR}을 출력하지만, 과선명화 (over sharpening) 효과로 인해 시각적 결함을 야기한다. 이러한 시각적 결함을 해결하기 위해 G의 크기를 증가하는 경우 하드웨어에 부적합 하며, 여러 가지의 손실 함수를 동시에 적용하는 경우 여러 손실 함수들의 장점들이 결합되는 동시에 단점도 함께 부각되는 문제가 있다.

대부분의 SISR 연구는 자연 영상 (natural image)들의 성능을 높이기 위해 진행 된다. 그러나, 실제 디스플레이 장치는 텍스트 영상과 자연 영상이 함께 출력 된다. 텍스트 영상의 경우 자연 영상들에 비해 시각적 결함에 취약하다. 텍스트 영상의 SISR은 자연 영상들에 비해 주목받지 않아 기계 학습 기반의 방법들[16, 17]과 SRCNN의 연구 결과가 존재한다. 그러나 기존 연구들은 학습용 영상과 실제 테스트 영상들의 배경과 폰트의 색상 조합이 동일하다. 기계 학습 기반의 방법들을 포함하여 CNN 기반의 텍스트 SISR은 학습 된 색상의 영상에 대해서 높은 성능을 보이지만, 학습되지 않은 영상을 테스트 하는 경우 심한 시각적 결함을 보인다. 텍스트 영상은 다양한 폰트와 배경의 색상 조합이 가능하며 모든 조합을 학습 하는 것을 불가능한 문제가 있다.

제 2 절 연구의 내용

본 논문은 SISR을 위한 CNN을 하드웨어 관점으로 최적화 하기 위한 세 가지 방법을 제안한다. 첫 번째는 스트리밍 구조 (streaming architecture) CNN 하드웨어 설계 방법이다. 스트리밍 구조는 off-chip 메모리 접근 없이 모든 연산을 하는 하드웨어 구조로 래스터 스캔 순서 (raster scan order)로 실시간으로 입력되는 영상에 대해서 실시간 연산이 필수인 디스플레이 장치에 적용되는 하드웨어 구조이다.

SISR용 CNN 하드웨어를 설계하기 위한 CNN 구조는 VDSR을 사용한다. VDSR은 모든 컨볼루션 레이어의 필터 크기가 3×3으로 동일하며 마지막 컨볼루션 레이어를 제외한 모든 컨볼루션 레이어들이 64개의 $fmap$ 을 출력한다. 이는 CNN의 컨볼루션 레이어의 개수와 output $fmap$ ($ofmap$) 수에 따른 SISR 성능 분석 및 하드웨어 설계에 용이한 장점이 있다. 입력 영상은 CNN 하드웨어로 래스터 스캔 순서로 입력되어 동일한 순서로 출력 된다. 따라서 컨볼루션 레이어로 입력된 이전 라인의 데이터들은 on-chip 메모리에 저장되며 이 on-chip 메모리는 $fmap$ 이 입력되는 매 사이클마다 read 및 write 동작이 동시에 발생한다. 동시에 read 및 write 동작을 만족하기 위해 on-chip 메모리는 듀얼 포트 SRAM (dual port SRAM) 으로 구성되거나 싱글 포트 (single port SRAM)을 사용하여 더블 버퍼링 (double buffering) 구조를 갖는다. 본 논문에서는 컨볼루션 레이어의 연산 순서를 래스터 스캔 순서에서 부분적 수직 순서 (partially-vertical order)로 변경하여 기존의 CNN 하드웨어에 적용되는 듀얼 포트 SRAM을 더블 버퍼링 없이 싱글 포트 SRAM으로 변경한다. 동일한 용량을 저장하기 위해 싱글 포트 SRAM은 듀얼 포트 SRAM과 비교하여 두 배의 면적을

자치한다. 따라서 제안하는 부분적 수직 순서 연산은 기존의 래스터 스캔 순서 기반의 CNN 하드웨어와 동일한 양의 $fmap$을 저장하면서도 절반의 면적을 차지한다.

CNN의 on-chip SRAM 개수는 필터의 높이에 비례한다. 그러나 VDSR은 모두 3×3 필터로 구성되므로 대칭 (symmetric) 형태의 필터 중 최소 크기이다. 따라서 VDSR의 on-chip SRAM의 개수를 감소하기 위한 필터의 형태를 변경 하는 방법을 제안한다. 필터 형태의 변경은 컨볼루션 레이어 병합, 컨텍스트 보존 (context-preserving) 1D 필터 구성 및 수직 방향 1D 필터 크기 감소로 나뉘어진다. 위의 세 단계를 거친 1D 컨볼루션 필터로 구성된 VDSR은 on-chip SRAM 개수를 절반으로 감소하면서도 SISR 성능 저하가 미미하여 on-chip SRAM 감소에 적합한 필터 구조이다. CNN 하드웨어 설계를 위한 위의 방법들 더불어 sub-pixel 컨볼루션 레이어를 포함한 하드웨어 리소스 사용량 감소 방법들은 디스플레이 장치에 적용이 가능한 스트리밍 구조의 CNN 하드웨어 설계를 가능하게 한다.

이미 설계된 하드웨어는 그 구조를 변경 하는 것이 불가능 하다. 따라서, CNN 하드웨어의 구조는 유지한 채 SISR 성능을 높이기 위한 CNN 학습 방법에 대한 연구가 필요하다. CNN이 원본 영상과 시각적으로 유사한 영상을 출력하기 위한 방법으로는 SRGAN이 존재한다. SRGAN은 super resolution residual network (SRResNet) [14] 구조의 SISR CNN를 픽셀 값 기반의 손실 함수로 학습한 경우와 VGG 네트워크의 $ofmap$ 기반의 손실 함수로 학습한 경우의 시각적 유사성의 차이가 있음을 보이며, 원본 영상과 SISR 결과 영상의 차이를 감소하기 위해 이를 판별하는 D 기반의 적대적 손실을 적용한 경우 원본 영상과 시각적으로 유사한 영상을 출력함을 보인다. 그러나

SRGAN은 과선명화 영향으로 인해 영상의 시각적 결함을 초래한다. 본 논문은 SRGAN의 시각적 결함의 원인으로 SRGAN의 G에 적용되는 손실 함수들이 영상의 세밀한 정보들을 제거함에 있음을 보인다. 더불어 영상의 세밀한 정보들을 유지하는 손실 함수를 적용하기 위한 D의 구조와 VGG 네트워크 구조 변경 방법을 제안한다. 제안하는 D의 구조와 VGG 네트워크 구조는 원본 영상의 세밀한 정보들을 유지하여 G로 하여금 원본 영상과 유사한 SISR 영상을 출력하도록 한다. 이는 이미 설계된 CNN 하드웨어의 재설계 없이 SISR의 성능을 향상한다.

기존의 SISR연구들은 자연 영상의 SISR에 집중되어 있으나, 텍스트 영상의 시각적 결함이 가독성을 저하함으로 더 큰 문제를 야기한다. 기존의 CNN 학습 방법들은 모델의 일반화(generalization)을 최대화 하기 위해 다양한 종류의 영상들을 학습한다. 그러나 텍스트 영상의 경우 배경과 폰트의 모든 색상 조합을 학습하는 것은 불가능 하다. 본 논문은 기존의 텍스트 영상 압축에 사용되는 방법 중 de-colorization [18] 을 CNN에 적용 한다. De-colorization은 원본 영상의 색상 정보 복원이 가능한 RGB to gray scale 변환 방법으로 CNN에 적용하는 경우 CNN은 입력 영상의 색상과 무관하게 검은색 폰트와 흰 색 배경으로 이루어진 영상의 SISR을 학습하게 되어 학습되지 않은 다양한 영상에 대해서도 시각적 결함 없는 SISR 영상 출력을 가능하게 한다.

제 3 절 논문의 구성

본 논문의 구성은 다음과 같다. 제 2장은 본 논문의 연구와 연관이 있는 SISR을 위한 CNN에 대한 기존 연구 소개를 한다. 제 3장은 VDSR을 기반으로 한 SISR용 CNN 하드웨어 설계를 위한 on-chip SRAM의 면적을 감소하기 위한 방법을 제안한다. 제 4장은 VDSR의 on-chip SRAM의 크기를 감소하기 위한 필터 구조 변경 방법과 이를 적용한 하드웨어 설계를 제안한다. 제 5장은 SISR용 CNN 하드웨어가 미리 설계된 경우 SISR 성능 개선을 위한 연구로 해상도 보존 구조 초해상도 적대적 생성 신경망 구조를 제안한다. 제 6장은 텍스트 영상의 SISR을 위한 de-colorization 을 제안하며 제 7장은 본 논문의 결론을 낸다.

제 2 장 이전 연구

2.1 SISR CNN 알고리즘

2.1.1 SRCNN

VDSR은 VGG network 구조를 기반으로 한 SISR용 CNN으로 20개의 컨볼루션 레이어로 구성 된다 [11]. 그림 2.1은 VDSR의 구조를 보인다. VDSR의 첫 번째 컨볼루션 레이어는 64개의 채널을 갖는 $ofmap$을 출력하며 VDSR의 마지막 컨볼루션 레이어는 영상의 Y 채널을 출력한다. VDSR은 큰 콘텍스트 크기와 skip connection으로 높은 SISR 성능을 보인다.

기존의 SISR용 CNN인 SRCNN [9], fast super-resolution convolution neural network (FSRCNN) [19] 및 efficient sub-pixel convolution neural network (ESPCN) [20] 의 콘텍스트 크기는 각각 17×17, 21×21 및 9×9이다. 반면 VDSR의 콘텍스트 크기는 41×41이다. VDSR의 큰 콘텍스트 크기는 특정 픽셀을 기준으로 주변의 많은 영상 정보들을 활용하여 SISR 연산을 수행하게 하여 영상의 세밀한 정보들을 복원 하는 것을 가능하게 한다.

VDSR 구조는 첫 번째 컨볼루션 레이어에 입력되는 영상 데이터와

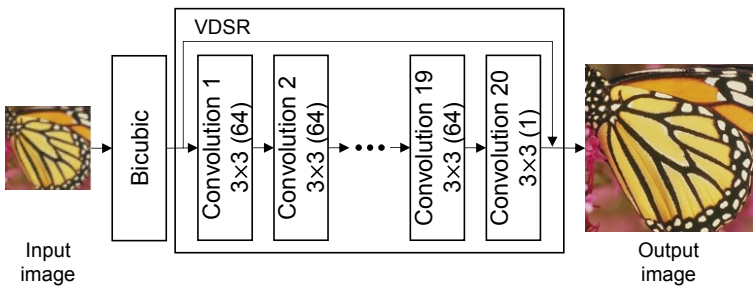


그림 2.1 VDSR의 구조

마지막 컨볼루션 레이어의 출력 데이터를 더하는 skip connection이 적용된다. 이는 CNN이 I^{HR} 과 I^{LR} 간의 차이인 residual 신호를 감소하도록 학습 되게 한다. I^{LR} 은 VDSR에 입력되기 전 Bicubic 보간법이 적용되어 I^{HR}과 동일한 해상도로 확대 되며, 이 확대된 영상이 VDSR에 입력되어 SISR 연산이 수행 된다.

기존의 SISR용 CNN구조들과 비교하여 VDSR은 컨볼루션 레이어의 개수를 조절하거나, 컨볼루션 레이어의 필터 수를 조절하기 용이한 구조를 갖는다. 따라서 하드웨어 구현에 적합한 구조의 CNN이다. 그러나, VDSR의 많은 컨볼루션 레이어들은 많은 on-chip 메모리를 사용하며 디스플레이 장치에 적용하기 위한 스트리밍 구조의 CNN 하드웨어를 설계하기 위하여 CNN의 곱셈기 개수와 on-chip 메모리 크기 감소 방법이 필요하다.

2.1.2 ESPCN

ESPCN은 SRCNN의 구조를 기반으로 한 SISR용 CNN 구조로 SRCNN의 소프트웨어 동작 속도를 향상하기 위한 방법을 제안 한다 [20]. ESPCN은 SRCNN과 동일하게 세 개의 컨볼루션 레이어로 구성되며 SRCNN의 재건 레이어 (reconstruction layer)의 역할을 수행하는 컨볼루션 레이어를 sub-pixel 컨볼루션 레이어로 변환한다. Sub-pixel 컨볼루션 레이어는 입력되는 input $fmap$ ($ifmap$)에 대해 super-resolution 확대 비율 (scale factor; sf)의 제곱인 sf^2채널의 $ofmap$을 출력 한다. 이 sf^2 채널의 $ofmap$ 은 수식 (2.1)에 따라 pixel-wise 배치가 되어 한 개 채널의 영상이 출력 된다.

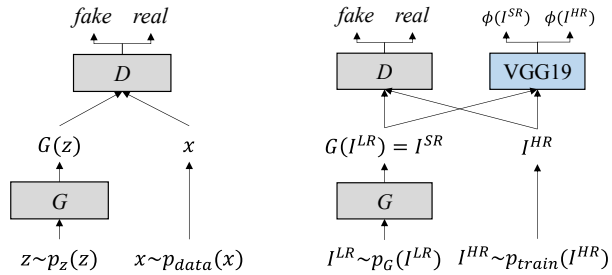
$$PS_{SR}(T)_{x,y} = T_{floor(\frac{x}{sf}),floor(\frac{y}{sf}),sf\times mod(y,sf)+mod(x,sf)} \quad (2.1)$$

수식 (2.1)의 $PS_{SR}(T)_{x,y}$는 sub-pixel 컨볼루션과 pixel-wise 배치 결과 (x,y) 에 위치하는 픽셀으로 sub-pixel 컨볼루션 레이어에서 출력되는 sf^2 채널의 텐서 (tensor; T)에 대해서 해당 수식이 적용된다.

SRCNN은 VDSR과 마찬가지로 I^{LR}에 Bicubic 보간법이 적용되어 super-resolution과 동일한 크기의 영상이 CNN에 입력 된다. ESPCN은 I^{LR} 에 Bicubic 보간법을 적용하지 않으며 CNN의 모든 컨볼루션 레이어들의 연산들은 I^{LR} 해상도의 영상을 기준으로 수행 된다. 따라서 SRCNN과 비교하여 컨볼루션 연산량이 $\frac{1}{sf^2}$ 로 감소하여 소프트웨어 동작 속도를 향상 시킨다. Sub-pixel 컨볼루션 레이어를 하드웨어에 적용하는 경우 CNN 하드웨어는 저해상도의 $fmap$을 on-chip 메모리에 저장하므로 CNN내 on-chip 메모리 크기를 $\frac{1}{sf}$로 감소할 수 있다.

2.1.3 SRGAN

SRGAN은 SISR에 generative adversarial networks (GANs) 구조를 적용한다 [14]. 그림 2.2는 기존의 GANs 구조와 SRGAN의 구조를 보인다. SRGAN은 판별자 네트워크 (D)와 생성자 네트워크 (G)로 구성 된다. SRGAN의 G는 I^{LR}을 입력 받아 I^{SR} $(=G(I^{LR}))$을 출력한다. D는 G로부터 출력되는 I^{SR}과 원본 영상인 I^{HR}을 입력 받아 각각의 영상에 대해서 fake와 real 판별을 한다. SRGAN의 D와 G는



(a) Original GANS (b) SRGAN

그림 2.2 기존 GANs 구조와 SRGAN 구조의 비교

GANs와 동일하게 two-player minimax game을 하며 가치 함수 (value function) $V(D,G)$는 수식 (2.2)와 같다.

$$\min_G \max_D V(D,G) = \mathbb{E}_{I^{HR}\sim P_{train}(I^{HR})}[\log D(I^{HR})] + \mathbb{E}_{I^{LR}\sim p_G(I^{LR})}\left[\log\left(1 - D\left(G(I^{LR})\right)\right)\right] \quad (2.2)$$

G는 perceptual loss (l^{SR})으로부터 학습 된다. l^{SR}은 적대적 손실 (adversarial loss; l^{SR}_{Gen})과 콘텐트 손실 (content loss; $l^{SR}_{VGG/i}$)로 구성된다. 콘텐트 손실은 GANs 구조에 새로 추가된 손실으로 VGG19 network의 i 번째 컨볼루션 레이어에서 출력되는 $ofmap$으로 계산되며 수식 (2.3)과 같다.

$$l^{SR}_{VGG/i} = \frac{1}{W_i H_i}\sum_{x=1}^{W_i}\sum_{y=1}^{H_i}\left(\phi_i(I^{HR})_{x,y} - \phi_i\left(G(I^{LR})\right)_{x,y}\right)^2 \quad (2.3)$$

수식 (2.3)의 ϕ_i는 i 번째 컨볼루션 레이어에서 출력되는 $ofmap$을 의미한다. W_i 및 H_i는 ϕ_i의 너비와 높이를 의미한다. SRGAN의 지각적 손실 (perceptual loss)는 수식 (2.4)를 따른다.

$$l^{SR} = l^{SR}_{VGG/i} + 10^{-3} l^{SR}_{Gen} \qquad (2.4)$$

SRGAN의 G는 I^{LR}에 대해서 I^{HR}과 동일한 I^{SR}을 생성하는 것을 목표로 하여 $l^{SR}_{VGG/i}$는 I^{HR}과 I^{SR}이 VGG19 네트워크의 $ofmap$ 도메인에서 동일해 지는 것을 목표로 하는 동시에 l^{SR}_{Gen}은 D가 I^{HR}으로 인식할 I^{SR}을 출력 하는 것을 목표로 한다.

기존의 SRGAN 구조는 시각적으로 원본과 유사한 I^{SR}을 출력하나 이 I^{SR}은 시각적인 결함 또한 갖고 있다. 이 시각적인 결함은 고주파 성분의 노이즈가 영상에 발생 하는 것으로 기존 연구들 중 G의 구조를 유지하면서 성능을 높이기 위한 방법으로는 여러 개의 손실 함수를 합치는 방법이 존재한다. [42]는 콘텐트 손실에 VGG19의 $ofmap$ 으로부터 발생하는 손실과 함께 영상간의 픽셀 차이를 함께 사용한다. 해당 방법은 고주파 성분의 노이즈를 감소하는 효과가 있는 동시에 SISR 결과 영상의 번짐 (blur) 현상이 발생한다. [43]의 연구는 VGG19의 $ofmap$ 과 이 $ofmap$의 Graham matrix를 적용한 손실 함수를 콘텐트 손실로 사용한다. 해당 방법은 VGG 네트워크를 기반으로 한 콘텐트 손실으로 노이즈를 효과적으로 제거하지 못하는 단점이 있다.

기존 연구들은 콘텐트 손실을 추가하여 고주파 성분의 노이즈를 제거하고자 하였다. 하지만 다수의 콘텐트 손실을 더하는 방법은 더해지는 콘텐트 손실의 장점과 함께 단점도 증가하는 문제가 존재한다. 고주파 성분 노이즈로 인한 시각적인 결함의 원인은 적대적 손실을 생성하는 D와 콘텐트 손실을 생성하는 VGG19 네트워크가 영상의

세밀한 정보를 손실하는 구조를 가짐에 있으며, 이는 SISR의 목적인 영상의 세밀한 정보를 유지하는 것과 반대로 동작 하는 것이다.

2.2 스트리밍 구조의 SISR 하드웨어

스트리밍 구조는 off-chip 메모리의 접근 없이 하드웨어의 중간 연산 결과들을 on-chip 메모리에 저장하는 구조이다. M.C. Yang *et al.* 은 anchored neighborhood regression (ANR) [7]를 기반으로 한 하드웨어 구조를 제안한다 [21]. 해당 구조는 $s=2$의 환경에서 Full-HD 해상도 영상을 60fps로 출력 한다. 그러나 CNN 기반의 SISR 방법들과 비교하여 해당 방법은 낮은 성능을 보인다.

CNN은 hidden 컨볼루션 레이어들의 $fmap$ 의 채널 수가 입력 영상에 비해 매우 많으며 SISR용 CNN은 고해상도의 $fmap$ 을 중간 결과로 출력하여 on-chip 메모리의 면적이 영상 분류용 CNN과 비교하여 매우 큰 특징이 있다. 따라서, 작은 네트워크 크기를 갖는 CNN들이 주로 SISR용 CNN 하드웨어에 적용 된다.

T. Manabe *et al.* 은 SRCNN을 기반으로 한 CNN 하드웨어 구조를 제안한다 [22]. T. Manabe *et al.*은 FPGA 의 제한적인 하드웨어 리소스를 해결하기 위한 방법을 제안 한다. SRCNN의 5×5 크기의 필터 대신 2×2 필터를 사용하며 총 4개의 컨볼루션 레이어로 구성된다. 하드웨어 리소스를 적게 사용하기 위해 4개의 컨볼루션 레이어는 각각 32, 16, 16, 1개 채널의 $ofmap$을 출력한다.

FSRCNN을 하드웨어 설계한 연구는 다음과 같다 [23, 24]. FSRCNN은 7개의 컨볼루션 레이어와 한 개의 디컨볼루션 (de-convolution) 레이어로 구성 된다. 7개의 컨볼루션 레이어 중 중간 파트

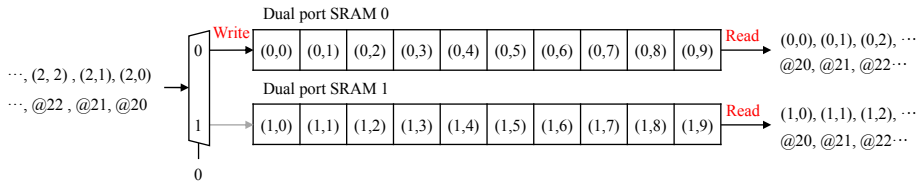


그림 2.3 듀얼 포트 SRAM 구조의 on-chip SRAM

(mid part)에 해당하는 여섯 개의 컨볼루션 레이어 중 다섯 개의 컨볼루션 레이어는 12개 채널의 $ofmap$을 출력하며 SRCNN과 비교하여 4.59배 작은 크기의 네트워크로 하드웨어 설계에 용이하다. [24] 는 FSRCNN의 디컨볼루션 레이어를 컨볼루션 레이어로 변경한 하드웨어 구조를 사용한다. [24]는 [23] 구조를 기반으로 컨볼루션 필터 크기를 감소하였으며, 각 컨볼루션 필터마다 출력 채널 개수를 감소하여 하드웨어 리소스 사용량을 감소하였으나, 큰 성능 저하를 야기한다.

2.3 기존 CNN 하드웨어의 on-chip 메모리 감소 방법

스트리밍 구조의 CNN 하드웨어는 래스터 스캔 순서로 $fmap$ 이 입력되며 같은 순서로 컨볼루션 레이어의 연산 결과가 출력된다. 따라서 그림 2.2와 같이 컨볼루션 필터에 사용할 $fmap$ 을 SRAM 으로부터 읽는 동작과 새로 컨볼루션 레이어에 입력되는 $ifmap$ 을 SRAM에 저장하는 동작이 함께 발생한다. 그림 2.2는 3×3 필터 연산을 위한 on-chip SRAM의 구조를 보인다. 사이클 20에 $fmap(2,0)$은 컨볼루션 레이어로 입력 된다. 같은 사이클에 SRAM0과 1로부터 각각 $fmap(0,0)$ 과 $fmap(1,0)$ 을 읽는 동작이 발생하며, $fmap(2,0)$ 은 SRAM0의 $fmap(0,0)$이 저장되었던 주소에 새로 저장된다.

싱글 포트 SRAM은 동일한 양의 데이터를 저장하는데 듀얼 포트

SRAM의 절반의 면적을 사용한다. 따라서, 듀얼 포트 SRAM의 크기를 절반으로 줄이는 것과 동일한 효과를 보인다. 그러나, SRAM의 읽기와 쓰기 동작이 동시에 수행될 수 없으며 기존의 CNN 하드웨어가 싱글 포트 SRAM을 사용하는 경우 $ifmap$이 컨볼루션 레이어에 입력되는 타이밍을 조절해야 하며 이는 스트리밍 구조의 하드웨어에 적용하는 것이 불가능 하다.

기존의 CNN 하드웨어 구조들은 on-chip 메모리 제약 문제를 해결하기 위하여 off-chip 메모리를 사용하여 메모리 계층 구조를 사용한다 [12, 13]. 그러나, 이러한 메모리 계층 구조는 off-chip 메모리 접근에 따른 latency에 의하여 실시간 동작이 불가능하며 디스플레이 장치를 위한 CNN 구조에 부적합 하다.

메모리 계층 구조를 사용하지 않는 on-chip 메모리 감소 방법은 다음과 같다. 첫 번째 방법으로는 CNN의 연산 순서를 변경하고 싱글 포트 SRAM을 적용하는 방법 이다 [38]. 이 구조는 기존의 래스터 스캔 순서와 달리 수직 방향으로 컨볼루션 연산을 수행한다. 기존의 래스터 스캔 순서로 연산을 하는 경우 SRAM의 크기는 입력 영상의 너비에 비례하지만 수직 방향으로 적용하는 해당 방법은 SRAM의 크기가 입력 영상의 높이에 비례한다. SISR 연산에 사용되는 일반적인 영상들은 영상의 너비가 높이에 비해 크므로 이는 SRAM 크기를 감소할 수 있다. 또한, 이 구조는 싱글 포트 SRAM을 사용한다. 싱글 포트 SRAM을 사용함으로 듀얼 포트 SRAM과 비교하여 SRAM 면적을 절반으로 감소할 수 있다. 그러나 해당 구조의 연산 방향은 단순히 입력 영상을 90° 회전 한 것과 동일하여 싱글 포트 사용시 입력 영상의 타이밍 컨트롤이 필요하고, 이는 스트리밍 구조에 적합하지 않다. 다른 방법으로는 SRAM에 저장되는 데이터의 양을 감소하기 위한 방법으로

$fmap$ 압축이 있다[40, 41]. [40]는 Huffman 코딩을 통하여 on-chip 메모리의 크기를 20~30% 압축하며 [41]은 자주 발생하는 $fmap$ 값을 테이블에 저장하여 SRAM의 크기를 감소한다. 하지만 해당 방법들은 고정적인 압축 아니므로 on-chip 메모리 면적을 감소하기에 효과적이지 않으며 부수적으로 압축 하드웨어 모듈과 압축을 해제하기 위한 하드웨어 모듈이 필요하다. 고정적으로 on-chip 메모리를 감소하기 위한 방법으로는 양자화(quantization)가 있다 [25, 39]. 그러나 해당 방법은 CNN의 성능을 저하하는 문제가 있다. 특히 SISR연산은 픽셀 값을 유지하는 것이 중요하며 양자화가 적용되는 경우 픽셀 값들이 손실되어 성능 하락을 야기할 수 있다.

2.4 De-colorization

De-colorization은 텍스트 영상을 압축하기 위해 제안 된 방법이다. 하나의 픽셀을 기준으로 픽셀 내 RGB 데이터를 sub-pixel이라 하며, 래스터 스캔 방향으로 RGB sub-pixel간의 gradient의 부호가 일정하게 한다. Gradient의 부호가 일정한 경우 텍스트 영상의 압축 효율을 높게 하는 장점이 있다. 텍스트 영상의 색상은 검은색 폰트와 흰색 배경을 기준으로 alpha blending [26] 과정을 거쳐 설정 된다. 수식 (2.5)는 텍스트 영상의 alpha blending 방법을 보인다.

$$C_{orig} = C_{font} + C_{decolor} \times (C_{bg} - C_{font})/255 \qquad (2,5)$$

수식 (2.5)의 C_{orig} 와 $C_{decolor}$ 는 원본 색상 영상과 de-colorized 텍스트 영상의 색상을 각각 의미한다. C_{font} 와 C_{bg} 는 각각 원본 색상

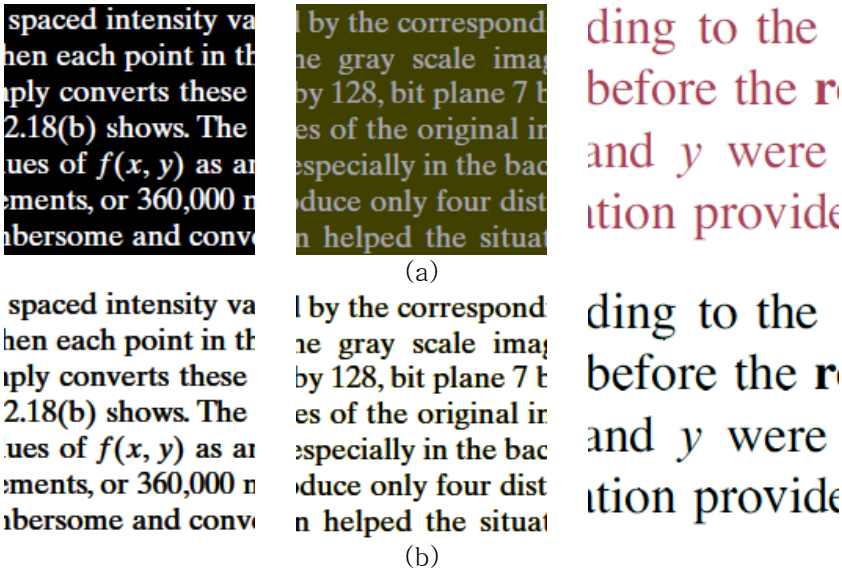


그림 2.4 다양한 배경과 텍스트의 색상 조합 영상에 대한 de-colorization 적용 결과. (a) 입력 영상; (b) de-colorization 적용 결과

영상의 폰트 색상과 배경 색상을 의미한다. De-colorization은 수식 (2.5)의 역함수로 수식 (2.6)과 같다.

$$C_{decolor} = 255 - 255 \times (C_{bg} - C_{orig})/(C_{bg} - C_{font}) \qquad (2.6)$$

그림 2.4은 다양한 폰트와 배경의 색상 조합을 갖는 텍스트 영상에 de-colorization을 적용한 결과를 보인다. 그림 2.3(a)는 원본 영상으로 영상 내 각각의 픽셀은 C_{orig} 의 색상 값을 갖는다. 그림 2.3(b)는 de-colorization이 적용된 영상으로 영상 내 각각의 픽셀이 $C_{decolor}$의 색상 값을 갖으며 de-colorization이 적용되어 검은 색 폰트와 흰색 배경의 색상 영상으로 변환 된다.

텍스트 영상의 색상을 제거하기 위한 방법으로는 RGB 영상을 gray 영상으로 변환하거나 YCbCr 채널 중 Y 채널만 사용하는 방법이

존재한다. 그러나 gray 영상은 색상 정보가 손실되어 SISR 결과 영상의 색상 정보 복원이 불가능 하다. 더불어 Y 채널만 사용하는 경우 색상 정보들은 Bicubic 보간법이 적용되며 이는 색상의 왜곡을 야기할 수 있다. 반면, de-colorization은 검은 색 폰트와 흰색 배경의 영상을 alpha blending 과정을 통해 색상 손실 없이 복원이 가능하다.

제 3 장 컨볼루션 뉴럴 네트워크의 SRAM 면적 감소를 위한 연산 순서 변경

본 장은 CNN 하드웨어 구조 내 사용되는 SRAM의 면적을 줄이기 위한 방법을 제안한다. 컨볼루션 레이어 내에서는 매 사이클 입력되는 $ifmap$ 데이터를 저장하기 위한 SRAM이 필요하며 기존의 컨볼루션 레이어 하드웨어 구조는 래스터 스캔 순서로 입력되는 데이터를 저장하기 위해 듀얼 포트 SRAM (dual port SRAM) 을 사용한다. 제안하는 방법은 컨볼루션 레이어의 연산 순서를 래스터 스캔 순서에서 부분적 수직 순서 (partially-vertical order)로 변경함으로 $ifmap$ SRAM을 듀얼 포트 SRAM을 싱글 포트 SRAM으로 변경하며 이로 인하여 SRAM의 면적을 감소한다. 제안하는 방법은 on-chip SRAM을 사용하는 CNN 가속기 구조에 적용이 가능하다.

3.1 부분적 수직 순서 컨볼루션 연산

동일한 양의 데이터를 SRAM에 저장하는 경우 듀얼 포트 SRAM은 싱글 포트 SRAM 대비 두 배의 면적을 사용한다. 본 장은 컨볼루션 레이어의 연산 순서를 변경함으로 SRAM의 쓰기 및 읽기 사이클이 동시에 발생하는 현상을 방지 한다. 그림 3.1은 부분적 수직 순서 컨볼루션 연산의 예시를 보인다. i 번째 컨볼루션 레이어에 입력되는 특징 맵의 높이 (L_i) 및 너비 (K_i) 는 각각 12와 5이며 필터의 높이 (H_i) 와 너비 (W_i) 는 모두 3이다. 또한, 이 예시는 제로 패딩 (zero padding) 을 고려한다. 그림 3.1(a)와 그림 3.1(b)의 상자 내 숫자는 연산 사이클을 의미하며 화살표는 연산 순서를 의미한다. 그림 3.1(c)는

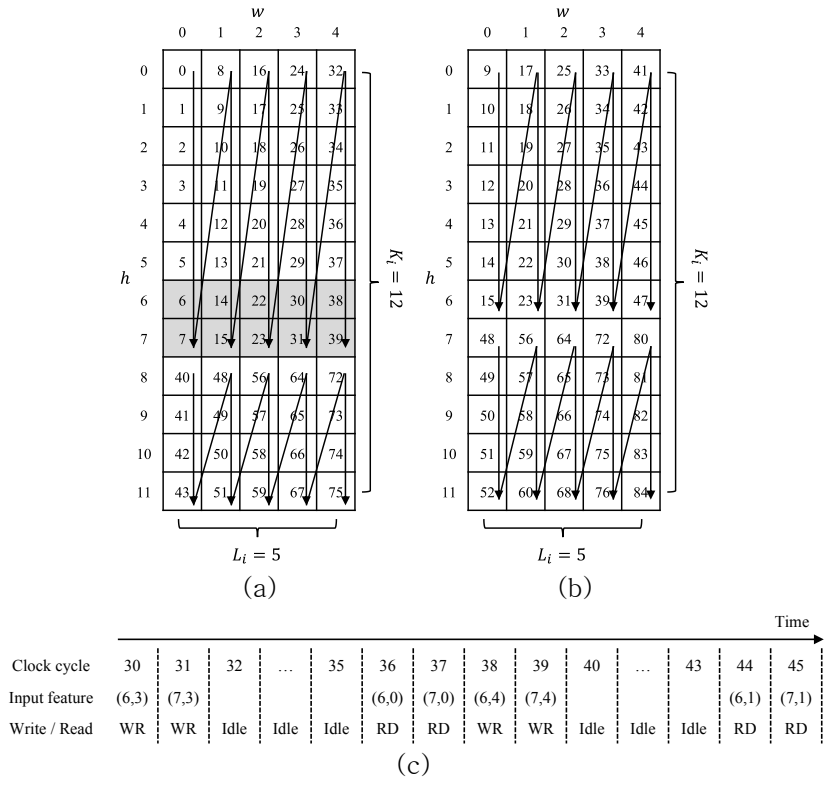


그림 3.1 부분적 수직 순서 컨볼루션 연산 예시. (a) $ifmap$의 컨볼루션 레이어 입력 순서; (b) 컨볼루션 연산 순서; (c) 클락 사이클에 대한 싱글 포트 SRAM의 쓰기 및 읽기 동작 예시

사이클 30부터 45까지의 SRAM의 동작을 나타낸다. 사이클 30 및 31에는 $ifmap$ (6,3)과 (7,3)이 SRAM에 저장 된다. 사이클 32에서 35 까지는 $ifmap$ (0,4)부터 (5,4)까지가 레지스터에 저장 되며 이 $ifmap$들은 SRAM에 저장되지 않는다. 사이클 36에는 SRAM으로부터 $ifmap$ (6,0)을 읽어서 사이클 37에 이 $ifmap$은 레지스터에 저장 된다. 사이클 38 및 39에는 $ifmap$ (6,4) 및 (7,4)가 SRAM으로부터 읽어져 레지스터에 저장 된다. 사이클 40부터 43까지는 매 사이클마다 $ifmap$ (8,0)부터 (11,0)까지가 컨볼루션 레이어에 입력되며 SRAM을 거치지 않고 레지스터에 저장 된다. 사이클 44 및 45에는 $ifmap$ (6,1)과 (7,1)이 SRAM으로부터 읽어져 레지스터에 한 사이클씩 지연되어 저장

Algorithm 2. 부분적 수직 연산 순서의 의사 코드 (batch size, M_i, and N_i are 1)

```
1   Partially-vertical convolution (ifmap, weight)
    Input: ifmap, weight
    Output: ofmap
2   For h=0; h<K_i; h+=pd_CONV do
3     For w=0; w<L_i; w++ do
4     // The partially-vertical order convolution
5       ofmap[h][w] = 0;
6       For h_temp=0; h_temp<pd_CONV; h_temp++ do
7         For j=0; j<H_i; j++ do
8           For k=0; k<W_i; k++ do
9             ofmap[h][w] += weight[j][k] *
10              ifmap[h+h_temp+j-(H_i-1)/2][w+k-(W_i-1)/2]
11          End
12        End
13      End
14    End
15  End
```

된다. 사이클 48에는 (7,0)을 중심으로 하는 3×3 필터 연산에 필요한 모든 $ifmap$ 들이 레지스터에 저장된 상태이므로 컨볼루션 연산을 수행한다. 이와 같이 사이클 49부터 52까지 $ifmap$ (8,0), (9,0), (10,0) 및 (11,0)을 중심으로 하는 컨볼루션 연산이 수행 된다. 이와 같이 $ifmap$ 의 0번째 열에 대해 부분적으로 컨볼루션 연산을 수행 한 후 사이클 56에 1번째 열에 대해 $ifmap$ (7,1)을 중심으로 하는 컨볼루션 연산을 수행 한다. 7번째 열에 대해서 컨볼루션 연산을 하기 위해서는 6, 7, 8번째 열의 $ifmap$ 이 필요하다. 그림 3.1의 예시는 하나의 행에 대해서 여덟 개의 열씩 $fmap$ 데이터가 컨볼루션 레이어에 입력 되므로 6, 7, 및 8번째 열의 $ifmap$ 을 사용하기 위해서는 그림 3.1(a)의 회색 상자와 와 같이 6 및 7번째 열의 $ifmap$ 데이터는 SRAM에 저장 되어야 한다.

부분적 수직 순서로 3 × 3 필터 연산을 하는 경우 싱글 포트 SRAM을 사용하기 위해서는 SRAM에 읽기 및 쓰기 동작이 서로 다른 사이클에 발생해야 한다. 그림 4.1은 매 행마다 8 회씩 부분적 수직 순서로 컨볼루션 연산을 수행한다 이 8회의 연산중 4 사이클을 싱글 포트 SRAM에 읽기 및 쓰기에 사용하며 4 사이클은 SRAM이 동작하지

않는다. 알고리즘 (Algorithm) 2 는 부분적 수직 연산 순서의 의사 코드(pseudo-code)를 보인다. 설명의 간략화를 위하여 한 개의 배치 및 컨볼루션 레이어의 필터 개수 (M_i) 와 입력 특징 맵의 채널 개수 (N_i)는 모두 1으로 가정 한다. 부분적 수직 연산 순서는 기존의 래스터 스캔 컨볼루션 순서와 다르게 h_temp 변수가 추가 된다. 이 h_temp 변수는 0부터 $pd_{CONV}-1$까지 증가하며 pd_{CONV}는 각각의 행에 대해서 부분적 수직 연산 순서를 수행할 열의 개수를 의미한다. 이 pd_{CONV}는 식 3.1에 의해서 정해진다.

$$pd_{CONV} = 2 \times (\max(H_i) - 1)_{for\ i \geq 1} \qquad (3.1)$$

식 3.1에 따르면 pd_{CONV}는 CNN의 첫 번째 컨볼루션 레이어를 제외한 나머지 모든 컨볼루션 레이어의 필터 높이에 따라 결정 된다. 앞서 그림 3.1과 같이 모든 컨볼루션 레이어의 필터 높이가 3인 경우 두 개의 열에 대한 데이터를 싱글 포트 SRAM에 쓰기 및 읽기를 하기 위해서는 각각의 행 마다 4 사이클이 필요 하다. 스트리밍 구조의 CNN 하드웨어는 모든 컨볼루션 레이어가 동일한 연산 순서를 갖으므로 CNN을 구성하는 컨볼루션 레이어 중 가장 필터의 높이가 큰 경우를 기준으로 하여 SRAM의 읽기 및 쓰기 동작이 동시에 발생하는 것을 방지 한다. 식 3.2는 부분적 수직 방향 연산 순서를 적용 시 각 컨볼루션 레이어마다 싱글 포트 SRAM에 쓰기 및 읽기 동작이 수행되는 사이클을 나타낸다. 식 3.2의 쓰기 및 읽기 사이클은 입력 특징 (0,0)이 입력 되는 사이클을 기준으로 상대적인 사이클을 의미한다. 부분적 수직 방향 연산 순서를 사용하는 경우 기존의 듀얼 포트 SRAM 기반의 컨볼루션 레이어와 동일하게 $H_i - 1$ 개의 열에 대한

$fmap$ 데이터를 저장 한다. 그러나 제안하는 방법은 싱글 포트 SRAM을 사용하므로 기존의 컨볼루션 레이어와 비교하여 절반의 면적으로 SRAM을 구성 한다.

$$\begin{cases} Write(h,w) = pd_{CONV} \times \left(L_i \times \left\lfloor \frac{h}{pd_{CONV}} \right\rfloor + w\right) + \mathrm{mod}(h, pd_{CONV}) \\ Read(h,w) = Write(h,w) + pd_{CONV} \times (L_i - 1) - (H_i - 1) \end{cases} \quad (3.2)$$

3.2 $ifmap$을 저장하기 위한 레지스터

각각의 컨볼루션 레이어 하드웨어는 필터 연산에 필요한 입력 특징 데이터를 레지스터에 임시적으로 저장한다. 레지스터에 저장되어 있는 $ifmap$들은 필터 연산을 하기 위한 곱셈기 및 덧셈기 (Multiplier and accumulator; MAC)에 전달되며, 이 MAC는 필터 연산을 수행 한다. 기존의 래스터 스캔 순서는 모든 입력 데이터를 듀얼 포트 SRAM에 저장 하므로 필터 크기와 동일한 크기의 레지스터를 사용한다. 반면 제안하는 부분적 수직 순서를 적용하는 경우 pd_{CONV} 개의 열 중 $H_i - 1$ 개의 열에 대한 $ifmap$ 만 SRAM에 저장하므로 나머지 $ifmap$ 은 레지스터에 저장 되어야 한다. 그림 3.2는 부분적 수직 순서를 적용한 컨볼루션 레이어의 $ifmap$ 레지스터의 동작을 보인다. 그림 3.2의 컨볼루션 필터의 H_i 와 W_i, $\max(H_i)$는 모두 3을 가정한다. 따라서 식 3.1을 적용하여 pd_{CONV} 는 4를 사용한다. 그림 4.2의 회색 상자들은 SRAM으로부터 읽어서 레지스터에 저장된 $ifmap$ 을 의미하며 이외의 상자들은 모두 이전 컨볼루션 레이어로부터 전달받은 $ifmap$ 이다. 컨볼루션 연산이 부분적 수직 순서로 수행되기 때문에 레지스터에 저장된 $ifmap$ 또한 동일한 순서로 불필요한 (outdated) 상태가 되며

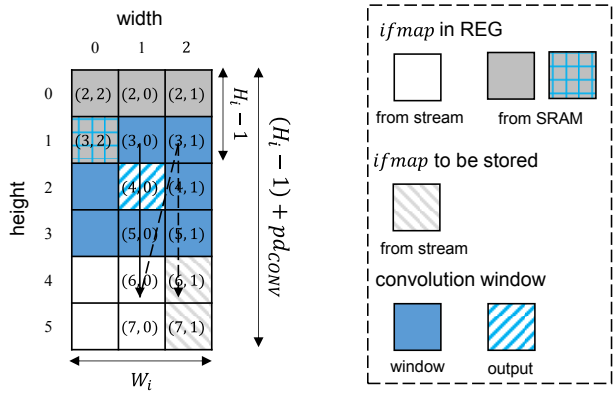


그림 3.2 부분적 수직 순서가 적용된 컨볼루션 레이어의 $ifmap$ 레지스터터

이 불필요해진 $ifmap$ 이 저장되어 있는 레지스터의 위치는 새로 입력되거나 SRAM으로부터 읽은 $ifmap$ 이 저장 된다. 그림 3.2에서 입력 특징 (4,0)을 중심으로 컨볼루션 연산을 수행하는 경우 레지스터의 (0,0)에 저장되어 있는 정보는 불필요한 상태이다. 따라서 $ifmap$ (2,2)를 SRAM으로부터 불러와 레지스터에 저장한다. $ifmap$ (4,0)에 대한 컨볼루션 연산 이후 $ifmap$ (5,0)을 중심으로 하는 컨볼루션 연산이 수행되며 레지스터 위치 (1,0)에 저장된 데이터는 불필요한 상태로 변경된다. 따라서 이 위치에는 $ifmap$ (3,2)가 저장되어 $ifmap$ (7,0)을 중심으로 하는 컨볼루션 연산 이후 $ifmap$ (4,0)을 중심으로 하는 컨볼루션 연산이 가능하게 한다. $ifmap$ (4,0)을 중심으로 하는 컨볼루션 연산시 $ifmap$ (6,1)이 이전 컨볼루션 레이어로부터 전달 된다. 이 $ifmap$ 은 SRAM에 저장되는 동시에 레지스터에도 저장된다. 이 $ifmap$ (6,1)은 입력 특징 (5,0)을 중심으로 하는 컨볼루션 연산에 활용되며 해당 연산이 발생하는 사이클에 $ifmap$

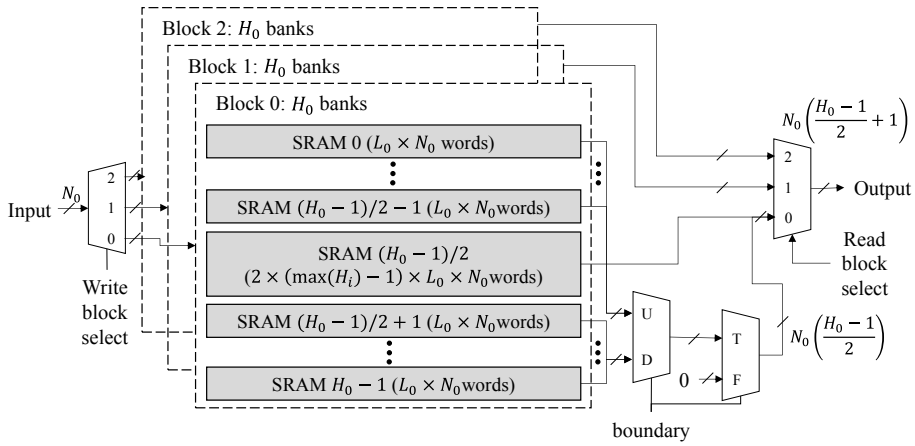


그림 3.3 부분적 수직 순서가 적용된 CNN의 첫 번째 컨볼루션 레이어의 SRAM 구조

(7,1)이 이전 컨볼루션 레이어로부터 전달되어 레지스터와 SRAM에 저장 된다. 식 3.3은 부분적 수직 순서를 적용한 컨볼루션 레이어의 $ifmap$ 레지스터 크기를 나타낸다.

$$\left((H_i - 1) + pd_{CONV}\right) \times W_i \times N_i \qquad (3.3)$$

3.3 CNN의 첫 번째 및 마지막 컨볼루션 레이어들의 SRAM 구성

일반적으로 입력 영상은 래스터 스캔 순서로 CNN에 입력 된다. 따라서 부분적 수직 순서로 컨볼루션 연산을 수행하기 위해서는 CNN의 첫 번째 컨볼루션 레이어에서 연산 순서를 변경하는 것이 필요하다. 해당 연산 순서는 SRAM을 사용하여 변경할 수 있으며 그림 3.3은 첫 번째 컨볼루션 레이어의 SRAM을 보인다. 그림 3.3의 메모리를 갖는 컨볼루션 레이어는 3×3 크기의 필터를 갖는 것을 가정한다. 첫 번째 컨볼루션 레이어의 SRAM은 세 개의 SRAM 블록으로 구성하며 모든

SRAM은 싱글 포트 SRAM이다. 하나의 SRAM 블록은 $H_0 - 1 + pd_{CONV}$개 라인의 입력 영상을 저장한다. 각각의 SRAM 블록의 윗 부분 및 바닥 부분의 $\frac{H_0-1}{2}$ 개의 SRAM 뱅크들은 첫 번째 컨볼루션 레이어의 필터 연산시 경계 데이터를 저장한다. 따라서 윗 부분과 바닥 부분의 $\frac{H_0-1}{2}$ 개의 SRAM 뱅크들은 동시에 동작 한다. 경계 부분 연산을 위한 SRAM 뱅크들을 구성하는 SRAM은 각각 $L_0 \times N_0$ 개의 데이터를 저장한다. 각 SRAM 블록의 SRAM $(H_0 - 1)/2$는 $pd_{CONV} \times L_0 \times N_0$ 개의 데이터를 저장하며 래스터 스캔 순서로 입력되는 영상 데이터를 메모리 주소 변경을 통해 부분적 수직 방향으로 출력 한다. 첫 번째 컨볼루션 레이어의 SRAM은 두 개 이상의 블록 들로 구성 된다. 수식 3.4는 부분적 수직 순서를 적용한 CNN의 첫 번째 컨볼루션 레이어의 SRAM 블록 수를 구하는 방법을 보인다.

$$\#\ blocks \geq \left\lceil \frac{(H_0-1+2\times pd_{CONV})\times L_0}{pd_{CONV}\times L_0} \right\rceil \qquad (3.4)$$

수식 3.4의 우 항의 분모는 SRAM의 읽기 및 쓰기에 필요한 사이클 수를 의미하며 분자는 읽기 동작에 필요한 클락 사이클 수를 의미한다. 싱글 포트 SRAM은 읽기 및 쓰기 포트를 공유하기 때문에 입력 영상을 SRAM에 저장하기 위해서 $(H_0 - 1 + 2 \times pd_{CONV}) \times L_0$ 사이클이 필요하며 SRAM에 저장된 데이터를 읽기 위해 $pd_{CONV} \times L_0$ 사이클이 필요하다. 따라서 한 개의 SRAM 블록은 $pd_{CONV} \times L_0$ 개의 데이터를 SRAM에 쓰기 위하여 $(H_0 - 1 + 2 \times pd_{CONV}) \times L_0$ 사이클이 필요하다. 한 개의 SRAM 블록을 사용하는 경우 SRAM에 쓰기 동작을 하는 사이클의 수가 SRAM으로부터 읽기 동작을 하는 사이클 수보다

크며 이는 메모리 메모리 오버플로 (memory overflow) 를 야기한다. 마지막 컨볼루션 레이어에서 출력된 CNN 연산 결과는 부분적 수직 순서로 출력 된다. 따라서 이 순서를 래스터 스캔 순서로 변경 해야 하며 이 또한 SRAM의 주소를 변경함으로 가능하다. 마지막 컨볼루션 레이어를 통과한 연산 결과는 추가적으로 컨볼루션 연산이 필요 없으므로 수식 3.4의 H_0 은 1이 된다. 따라서 이 SRAM은 두 개의 SRAM 블록으로 구성 된다.

3.4 $fmap$의 SRAM 다채널 공유를 위한 부분적 수직 순서 방법 적용

디스플레이 장치는 CNN에 한 라인에 해당하는 영상을 입력한 후 다음 라인 데이터를 CNN에 전달하기 전에 idle 사이클이 존재한다. 이 idle 사이클을 활용하여 각각의 $ifmap$ 채널마다 분리되어 있던 $ifmap$ SRAM을 서로 다른 채널의 $ifmap$이 공유할 수 있으며, 이는 $ifmap$ SRAM의 크기를 감소 할 수 있다. 서로 다른 채널의 $ifmap$을 같은 SRAM에 저장하기 위해서는 CNN의 첫 번째 컨볼루션 레이어의 연산 결과 $ofmap$의 채널이 분리되어 서로 다른 사이클에 다음 컨볼루션 레이어로 전달되어야 한다. 그림 3.4는 $ifmap$ SRAM이 두 채널의 $ifmap$을 저장하기 위해 첫 번째 컨볼루션 레이어의 연산 과정을 보인다. 그림 3.4에 입력되는 영상의 높이 (K_0) 와 너비 (L_0) 는 각각 12와 5이다. 배치 크기는 1이며 입력 영상의 채널 수 (N_0) 는 1이며 필터 개수 (M_0) 은 2이다. 그림 4.1과 동일하게 필터의 높이 (H_0) 및 너비 (W_0) 는 모두 3 이다. 그림 3.4(a)와 3.4(b)는 각각 컨볼루션 레이어에 영상이 입력되는 순서와 컨볼루션 결과 두 채널이 출력되는 순서를

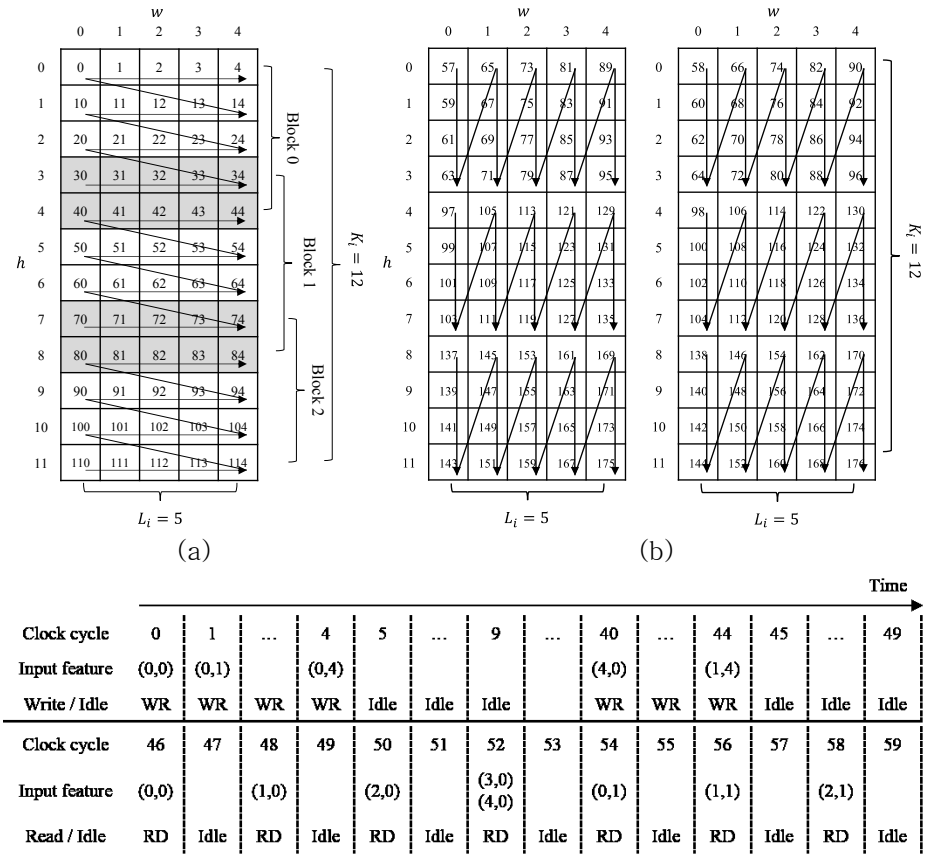
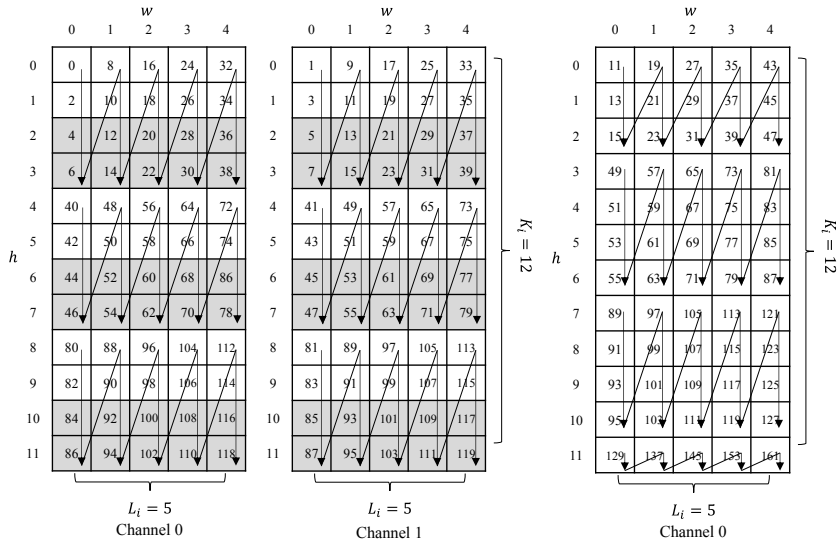


그림 3.4 $ifmap$ SRAM의 2 채널 공유를 위한 첫 번째 컨볼루션 레이어의 부분적 수직 순서 컨볼루션 연산 예시. (a) $ifmap$의 컨볼루션 레이어 입력 순서; (b) 컨볼루션 연산 순서; (c) 클락 사이클에 대한 싱글 포트 SRAM의 쓰기 및 읽기 동작 예시

보인다. 그림 3.4(c)는 사이클 0부터 59까지의 $ifmap$ SRAM의 쓰기 및 읽기 동작을 보인다. CNN에 입력되는 영상은 그림 3.1(a)와 같이 래스터 스캔 순서로 입력 된다. 0 번째 열이 CNN에 입력된 이후 5 사이클 동안은 영상이 CNN에 입력되지 않으며, 사이클 10부터 14까지 1 번째 열의 영상 데이터가 입력 된다. 이와 같이 영상의 한 개의 열은 영상의 너비의 두 배 사이클을 주기로 입력 된다. 3×3 컨볼루션 필터 연산의 경계 부분의 연산들을 고려하여 한 개의 SRAM 블록은 6 개의

라인 데이터를 저장 한다. 따라서 첫 번째 컨볼루션 레이어의 $ifmap$ SRAM의 block 0 은 line 0부터 line 4까지 저장한다. Line −1은 존재하지 않는 데이터로 제로 패딩을 적용 한다. 그림 3.4(a)의 회색 상자로 표시되어 있는 부분인 Line 3 및 line 4는 block 0과 block 1에 중복되어 저장 되며, 마찬가지로 line 7 및 line 8은 block 1과 block 2에 중복되어 저장 된다. 그림 3.4(c)와 같이 사이클 0부터 49 사이에 $ifmap$의 (0,0)부터 (4,4)가 $ifmap$ SRAM에 저장 된다. 이 50 사이클 동안 25 사이클은 idle 사이클이며 이는 한 픽셀이 2 사이클 마다 CNN에 입력되는 것과 처리량 (throughput) 이 동일하다. 사이클 46부터 59까지 $ifmap$의 (0,0)부터 (2,1)이 부분적 수직 순서로 $ifmap$ SRAM으로부터 읽어진다. 이 14 사이클 동안 $ifmap$ SRAM은 읽기와 idle 사이클이 반복되어 영상이 CNN에 입력되는 처리량과 동일하게 한다. 사이클 46부터 55까지 $ifmap$ (0,0)부터 (0,1)까지 SRAM으로부터 읽어져서 $ifmap$ REG에 저장 된다. 사이클 56에 $ifmap$ (1,1)이 SRAM으로부터 읽어져서 $ifmap$ REG에 사이클 57에 저장된다. 사이클 57에는 $ifmap$ (0,0), (1,0), (0,1) 과 (1,1)이 준비된 상태로 (0,0)을 중심으로 하는 $ofmap$에 대한 컨볼루션 연산이 수행 된다. 이 컨볼루션 연산은 $ofmap$ (0,0) 위치의 채널 0인 $(0,0)_0$과 채널 1인 $(0,0)_1$을 출력 한다. 이 $(0,0)_0$ 과 $(0,1)_1$ 사이에 지연 (delay) 사이클을 1 사이클 추가하여 $ofmap$의 채널을 분리하여 다음 컨볼루션 레이어에 전달한다.

그림 3.5는 CNN의 첫 번째 컨볼루션 레이어를 제외한 나머지 컨볼루션 레이어들에서 두 개의 채널이 하나의 $ifmap$ SRAM을 공유하는 예시를 보인다. 그림 3.5(a)와 그림 3.5(b)는 컨볼루션 레이어에 입력되는 $ifmap$과 컨볼루션 레이어에서 출력되는 $ofmap$ 의



(a) (b)

Time

Clock cycle	32	33	34	35	36	37	38	39	40	41	42	43	44	45
Input feature	(2,0)	(2,0)	(3,0)	(3,0)	(2,4)	(2,4)	(3,4)	(3,4)	(2,1)	(2,1)	(3,1)	(3,1)	(6,0)	(6,0)
Channel	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Write / Read	RD	RD	RD	RD	WR	WR	WR	WR	RD	RD	RD	RD	WR	WR

Clock cycle	46	47	48	49	50	51	52	53	54	55	56	57	58	59
Input feature	(7,0)	(7,0)	(2,2)	(2,2)	(3,2)	(3,2)	(6,1)	(6,1)	(7,1)	(7,1)	(2,3)	(2,3)	(3,3)	(3,3)
Channel	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Write / Read	WR	WR	RD	RD	RD	RD	WR	WR	WR	WR	RD	RD	RD	RD

(c)

그림 3.5 $ifmap$ SRAM의 2 채널 공유를 위한 컨볼루션 레이어의 부분적 수직 순서 컨볼루션 연산 예시. (a) $ifmap$의 컨볼루션 레이어 입력 순서; (b) 컨볼루션 연산 순서; (c) 클락 사이클에 대한 싱글 포트 SRAM의 쓰기 및 읽기 동작 예시

연산 순서를 보인다. 그림 3.5(c)는 사이클 32 부터 59 사이의 $ifmap$ SRAM의 동작을 보인다. 그림 3.5의 K_i, L_i, H_i, W_i, M_i와 배치 크기는 그림 3.4과 동일하다. 그림 3.5의 $ifmap$ 채널 개수 N_i는 2이다. (3,0)을 중심으로 3×3 컨볼루션 연산하는 과정은 다음과 같다. 사이클 32 및 33에는 $ifmap$ $(2,0)_0$ 및 $(2,0)_1$이 SRAM으로부터 읽어져서 각각 1 사이클씩 뒤 $ifmap$ REG에 저장 된다. 그리고 사이클 34 및 35에는

$ifmap$ $(3,0)_0$과 $(3,0)_1$이 이전 컨볼루션 레이어로부터 전달되어 $ifmap$ REG에 저장 된다. 사이클 36부터 39까지는 $ifmap$ $(2,4)_0$, $(2,4)_1$, $(3,4)_0$, $(3,4)_1$가 이전 컨볼루션 레이어로부터 전달되며 $ifmap$ SRAM에 저장된다. 이 입력 특징들은 그림 3.5(a)에 회색으로 표현되어 있다. 사이클 40부터 43까지 $ifmap$ $(2,1)_0$, $(2,1)_1$, $(3,1)_0$ 과 $(3,1)_1$은 $ifmap$ SRAM으로부터 읽어져 각각 1 사이클씩 후에 $ifmap$ REG에 저장된다. 사이클 44 부터 47은 $ifmap$ $(6,0)_0$, $(6,0)_1$, $(7,0)_0$ 과 $(7,0)_1$이 컨볼루션 레이어에 입력되며 이 특징들은 $ifmap$ SRAM 및 $ifmap$ REG에 저장된다. 사이클 48 및 49에 $ifmap$ $(4,1)_0$ 및 $(4,1)_1$ 이 입력되어 $ifmap$ REG에 저장된다. 사이클 49에는 $ifmap$ (3,0)을 중심으로 하는 3×3 컨볼루션 연산을 위한 $ifmap$ 데이터가 모두 준비되므로 컨볼루션 연산이 수행 된다. 컨볼루션 연산 결과인 $ofmap$ 의 출력 순서를 $ifmap$ 의 입력 순서와 동일하게 하기 위해 $ofmap$ $(3,0)_1$ 은 1 사이클 지연 후 다음 컨볼루션 레이어로 전달 된다. 따라서 사이클 50 및 51에 $(3,0)_0$ 및 $(3,0)_1$ 이 다음 컨볼루션 레이어로 전달 된다. 부분적 수직 순서로 컨볼루션 연산이 수행 되며 사이클 51, 53 과 55에 $ofmap$ (4,0), (5,0) 및 (6,0)이 계산된다. 각각의 $ofmap$ 의 채널 1은 채널 0보다 1 사이클씩 지연되어 다음 컨볼루션 레이어로 전달 된다.

부분적 수직 순서 컨볼루션 연산을 적용하여 두 개 이상의 채널이 하나의 $ifmap$ 의 SRAM을 공유하는 경우 식 3.5는 $ifmap$ $(h,w)_{Ch}$ 에 대해서 $ifmap$ SRAM의 쓰기 및 읽기 사이클을 나타낸다. 이 식에서 Ch 는 $ifmap$ 채널을 의미하며 그림 3.5의 예시에서는 0 또는 1이다. N^S 는 하나의 $ifmap$ SRAM을 공유하는 $ifmap$ 채널의 개수로 이 예시에서는 2 이다. pd_{CONV}^S 및 L_i^S는 pd_{CONV}와 L_i에 N^S를 곱한 값이다. h_{Ch}는 $N^S \times h + Ch$이다.

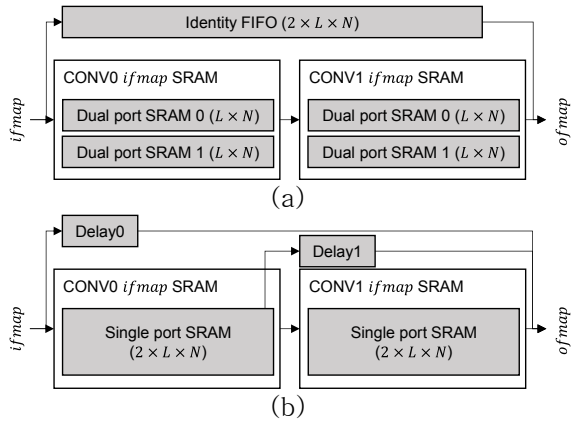


그림 3.6 Residual block에의 부분적 수직 순서 적용. (a) 기존 residual block 구조; (b) 부분적 수직 순서 적용 residual block 구조

$$\begin{cases} Write(h,w)_{Ch} = pd_{CONV}^{S} \times \left(L_i^S \times \left\lfloor \frac{h_{Ch}}{pd_{CONV}^{S}} \right\rfloor + w \right) + \mathrm{mod}\left(h_{Ch}, pd_{CONV}^{S}\right) \\ Read(h,w)_{Ch} = Write(h,w)_{Ch} + pd_{CONV}^{S} \times (L_i - 1) - N^S \times (H_i - 1) \end{cases}$$

(3.5)

마지막 컨볼루션 레이어 연산 결과인 $ofmap$은 지연 사이클 없이 바로 출력 된다. 따라서 N^S 사이클 마다 하나의 $ofmap$ 데이터가 출력되며 N^S가 2 이상인 경우 idle 사이클을 활용하여 래스터 스캔 순서로 변경하기 위한 SRAM 뱅크의 수를 한 개로 감소한다.

3.5 부분적 수직 순서의 적용 가능 CNN 구조

본 장은 CNN에 부분적 수직 순서를 적용 가능한 범위를 보인다. 부분적 수직 순서는 컨볼루션 레이어의 스트라이드 (stride) 크기와 무관하게 적용할 수 있다. 더불어 스트라이드가 적용된 컨볼루션 레이어로부터 출력되는 $ofmap$은 throughput이 1/stride^{2}로 감소한다.

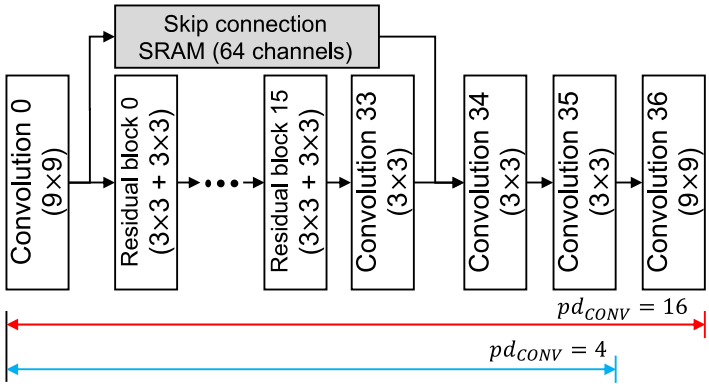


그림 3.7 SRResNet에의 부분적 수직 순서 적용.

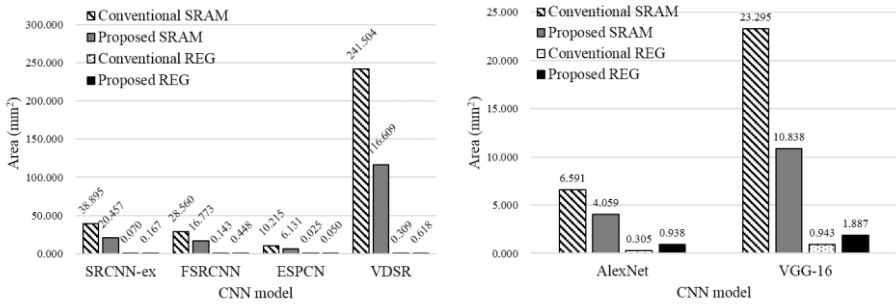
이는 한 개의 on-chip 메모리에 stride^{2}개 채널의 $ifmap$을 저장하여 SRAM 면적을 추가로 감소할 수 있다. SRGAN의 G구조인 SRResNet은 3×3 컨볼루션 필터로 구성되는 두 개의 컨볼루션 레이어로 residual block을 구성한다. 그림 3.6은 기존의 래스터 스캔 순서로 컨볼루션 연산을 처리하는 경우와 부분적 수직 순서를 적용하는 경우 on-chip SRAM 사용량을 보인다. 그림 3.6(a)의 기존의 residual block을 구성하는 각각의 컨볼루션 레이어는 $ifmap$ 채널 당 $H-1$개의 듀얼 포트 on-chip SRAM을 사용한다. 더불어 residual block의 연산 결과와의 덧셈 연산을 위해 identity $fmap$을 저장하는 identity FIFO를 사용한다. Identity FIFO는 $2 \times \frac{H-1}{2}$ 크기의 듀얼 포트 SRAM을 사용한다. 그림 3.6(b)는 residual block에 부분적 수직 순서를 적용한 하드웨어 구조를 보인다. 각각의 컨볼루션 레이어는 $ifmap$ 채널 당 1개의 싱글 포트 SRAM을 사용한다. 한 개의 싱글 포트 SRAM은 $H-1$ 라인의 $ifmap$을 저장한다. 부분적 수직 순서가 적용된 residual block은 입력되는 identity $fmap$에 일정 cycles의 delay를 주어 residual block의 연산 결과와 덧셈을 한다. 그림 3.6(b)의 delay0과

delay1은 각각 $2 \times (pd_{CONV} + H - 1)$ cycles와 $2 \times (pd_{CONV} + H)$ cycles 이다. pd_{CONV}는 영상의 너비인 L과 비교하여 매우 작기 때문에 residual block은 부분적 수직 순서를 적용한 경우 on-chip SRAM 면적이 $\frac{1}{3}$으로 감소한다.

부분적 수직 순서는 hidden 컨볼루션 레이어간 skip connection이 없는 CNN의 구조는 모두 적용 가능하다. 그러나, 3개 이상의 hidden 컨볼루션 레이어 사이에 skip connection이 있는 CNN의 구조에서는 pd_{CONV}를 최소화 하여 적용 한다. 그림 3.7은 [14]의 SRResNet의 구조를 보인다. SRResNet은 convolution 0과 convolution 36을 제외하고 모든 컨볼루션 레이어의 필터 크기가 3×3이다. 그러나 convolution 36이 9×9 필터를 사용함으로 pd_{CONV}는 16 cycles이다. 부분적 수직 순서를 사용하는 경우 skip connection 영역에 입력되는 $ifmap$을 저장하기 위한 skip connection SRAM의 크기는 수식 3.5와 같다.

$$L \times \left(pd_{CONV} - \frac{H-1}{2}\right) + \left(1 + \left\lfloor \frac{N_{layers}}{4} \right\rfloor\right) + \alpha \quad (3.5)$$

수식 3.5의 N_{layers}는 skip connection이 적용되는 컨볼루션 레이어의 개수이다. α는 컨볼루션 레이어 연산을 위해 소요되는 사이클의 수로 $(N_{layers} + 1) \times (pd_{CONV} + 1) \times \frac{H-1}{2}$ 이다. Skip connection SRAM은 dual port SRAM으로 구성되며, 이 SRAM의 크기는 수식 3.5와 같이 pd_{CONV}에 비례한다. 따라서 SRResNet 전체에 부분적 수직 순서를 적용하는 경우와 convolution 35까지 부분적 수직 순서를 적용하는 경우 pd_{CONV}는 4배 차이가 존재한다. 따라서, hidden 컨볼루션



(a) (b)

그림 3.8 SRAM 및 레지스터의 면적 비교 (a) SISR용 CNN (b) 영상 분류용 CNN

레이어에 skip connection을 적용하는 경우 pd_{CONV} 를 최소화 하여 부분적 수직 순서를 적용 한다.

3.6 실험 결과

본 장은 제안하는 부분적 수직 순서가 적용된 CNN 하드웨어의 SRAM 면적 감소를 실험적으로 확인한다. 면적을 계산하기 위한 방법으로 레지스터는 65-nm 공정으로 합성하였으며, on-chip 메모리인 SRAM의 면적은 an enhanced cache access and cycle time model (CACTI) 시뮬레이터를 사용 한다 [34]. SRAM 면적 변화를 확인하기 위한 CNN 구조들은 super-resolution을 위한 CNN인 SRCNN, FSRCNN, ESPCN 및 VDSR을 사용한다 [9, 19, 20, 11]. VDSR은 skip connection이 적용된 CNN 구조로 해당 실험은 skip connection을 위해 사용되는 SRAM은 제외하고 면적을 계산한다. 더불어 SISR을 위한 CNN은 Ultra-HD 영상을 출력하는 구조로 Y 채널만 연산에 사용되는 구조로 가정한다. SISR용 CNN 이외에 image classification을 위한 CNN으로 AlexNet [28]과 VGG network

[10]에도 부분적 수직 순서를 적용하여 SRAM 면적 감소를 확인 한다. AlexNet과 VGG network의 입력 영상 해상도는 512×512로 RGB 채널이 CNN으로 입력 된다. 모든 실험에 대해 입력 영상 및 feature map은 16-bit 부동소수점을 사용하는 것을 가정하며 batch의 크기는 1이다.

그림 3.8의 (a)와 (b)는 각각 SISR과 영상 분류용 CNN의 $ifmap$을 저장하기 위한 SRAM과 register의 면적을 보인다. 그림 3.8 (a)에서 제안하는 부분적 수직 방향 연산은 네 개의 CNN에 대해 기존 방법과 비교하여 SRAM의 면적을 1.85배 감소한다. 레지스터의 면적은 2.30배 증가하나 이 증가한 레지스터의 면적은 감소한 SRAM 면적과 비교하여 117.95배 작으므로 레지스터 면적의 증가는 무시할 수 있다.

표 3.1은 SISR용 CNN 하드웨어의 SRAM 및 레지스터 면적의 자세한 결과를 보인다. 부분적 수직 방향 연산 순서가 적용된 경우 hidden 컨볼루션 레이어의 SRAM 평균 면적은 2.11배 감소한다. Front 컨볼루

표 3.1 SISR용 CNN 하드웨어의 SRAM 및 레지스터 면적 (s=2)

Network	SRCNN-ex				FSRCNN			
	Conventional		Proposed		Conventional		Proposed	
	SRAM	REG	SRAM	REG	SRAM	REG	SRAM	REG
Front layer	0.794	0.002	2.249	0.004	0.208	0.001	1.362	0.003
Hidden layers	25.401	0.045	11.670	0.108	5.003	0.014	2.390	0.104
Last layer	12.700	0.023	5.835	0.054	23.348	0.128	10.211	0.341
rasterizer	-	-	0.703	-	-	-	2.810	-
Total Area/ Compression Ratio	38.895 /1.00	0.070 /1.00	20.457 /1.90	0.166 /0.42	28.560 /1.00	0.143 /1.00	16.773 /1.70	0.448 /0.32

Network	ESPCN				VDSR			
	Conventional		Proposed		Conventional		Proposed	
	SRAM	REG	SRAM	REG	SRAM	REG	SRAM	REG
Front layer	0.208	0.001	0.592	0.001	0.198	0.000	0.846	0.001
Hidden layers	6.671	0.016	3.186	0.033	228.605	0.293	109.325	0.585
Last layer	3.335	0.008	1.593	0.016	12.700	0.016	6.074	0.033
rasterizer	-	-	0.759	-	-	-	0.365	-
Total Area/ Compression Ratio	10.215 /1.00	0.025 /1.00	6.131 /1.67	0.050 /0.50	241.504 /1.00	0.309 /1.00	116.609 /2.07	0.619 /0.50

션 레이어의 평균 면적은 4.12배 증가하며 새로 추가된 rasterizer 모듈은 SRAM 면적을 증가시킨다. 그러나 front 컨볼루션 레이어 및 rasterizer 모듈은 전체 SRAM 면적의 1/27.96을 차지 하므로 이 부분에서 증가한 면적보다 hidden 컨볼루션 레이어에서 감소한 면적이 더 많다.

그림 3.8 (b)는 영상 분류용 CNN인 AlexNet과 VGG16 network의 SRAM 및 레지스터의 면적을 보인다. AlexNet과 VGG16 네트워크의 SRAM 면적은 각각 1.62배 및 2.20배 감소한다. 그러나 레지스터의 면적은 AlexNet과 VGG16 network에 대해 각각 3.07배와 2.00배 증가한다. 그러나 super-resolution용 CNN과 같이 증가한 레지스터의 면적은 감소한 SRAM 면적과 비교하여 작다.

표 3.2는 AlexNet과 VGG network에 대해서 자세한 면적 비교 결과를 보인다. AlexNet과 VGG network의 hidden 컨볼루션 레이어의 SRAM 평균 면적은 2.33배 감소한다. Front 컨볼루션 레이어의 면적은 기존의 구조와 비교해서 5.23배 증가한다. 그러나 해당 면적은 전체 CNN과 비교하여 15.59배 작은 면적이다. 레지스터의 평균 면적은 2.41배 증가하나 SRAM 전체 면적의 1/5.04배이다. 이 결과는 제안하는 부분적 수직 방향 컨볼루션 방법이 SISR과 같이 $fmap$의 해상도가 감소하지 않는 CNN구조에 높은 효율을 보이나 영상 분류를 위한 CNN

표 3.2 AlexNet과 VGG16 network의 SRAM 및 레지스터 면적

Network	AlexNet				VGG-16			
	Conventional		Proposed		Conventional		Proposed	
	SRAM	REG	SRAM	REG	SRAM	REG	SRAM	REG
Front layer	0.430	0.010	1.481	0.017	0.086	0.001	0.373	0.002
Hidden layers	4.873	0.230	2.035	0.704	21.762	0.813	9.624	1.625
Last layer	1.288	0.065	0.543	0.217	1.447	0.130	0.611	0.260
Total Area/ Compression Ratio	6.591 /1.00	0.305 /1.00	4.059 /1.62	0.938 /0.33	23.295 /1.00	0.943 /1.00	10.607 /2.20	1.887 /0.50

구조들에도 적용이 가능함을 보인다.

표 3.3은 SISR용 CNN내 하나의 SRAM에 여러 개의 채널의 $fmap$ 저장되는 경우 SRAM 면적 변화를 보인다. SRAM이 여러 채널의 $fmap$ 을 저장하는 경우에도 레지스터 면적은 변하지 않는다. 표 3.3의 N^S는 하나의 SRAM에 저장되는 $fmap$의 채널 개수를 의미한다. 따라서 N^S가 1인 경우 부분적 수직 방향 컨볼루션만 적용된 것과 동일하며 N^S가 2인 경우와 4인 경우는 각각 2개의 $fmap$채널과 4개의 $fmap$채널이 하나의 SRAM에 저장 되는 경우를 의미한다. N^S가 2인 경우와 4인 경우 SRAM의 면적은 각각 1.08배, 1.14배 감소한다. 따라서 여러 개의 채널이 동일한 SRAM에 저장되는 경우 SRAM의 면적이 감소하여 효율적인

표 3.3 SISR용 CNN의 SRAM이 여러 채널의 $ifmap$을 저장하는 경우 SRAM 면적

Network	SRCNN-ex			FSRCNN		
N^S	1	2	4	1	2	4
Front layer	2.249	2.249	2.249	1.362	1.362	1.362
Hidden layers	11.67	11.242	10.921	2.39	2.278	2.188
Last layer	5.835	5.621	5.461	10.211	9.831	9.556
Rasterizer	0.703	0.351	0.176	2.81	1.405	0.703
Total area/ Compression Ratio	20.457 /1.00	19.463 /1.05	18.806 /1.09	16.773 /1.00	14.881 /1.13	13.808 /1.21
Network	ESPCN			VDSR		
N^S	1	2	1	2	1	2
Front layer	0.592	0.592	0.592	0.592	0.592	0.592
Hidden layers	3.186	3.037	3.186	3.037	3.186	3.037
Last layer	1.593	1.518	1.593	1.518	1.593	1.518
Rasterizer	0.759	0.38	0.759	0.38	0.759	0.38
Total area/ Compression Ratio	6.131 /1.00	5.527 /1.11	6.131 /1.00	5.527 /1.11	6.131 /1.00	5.527 /1.11

표 3.4 AlexNet 및 VGG16 network의 SRAM이 여러 채널의 $ifmap$을 저장하는 경우 SRAM 면적

Network	AlexNet			VGG-16		
N^S	1	2	4	1	2	4
Front layer	1.481	1.481	1.481	0.373	0.373	0.373
Hidden layers	2.035	1.944	1.783	9.624	8.955	8.673
Last layer	0.543	0.493	0.451	0.611	0.543	0.493
Total area/ Compression Ratio	3.591 /1.00	3.918 /1.04	3.715 /1.09	10.607 /1.00	9.871 /1.07	9.539 /1.11

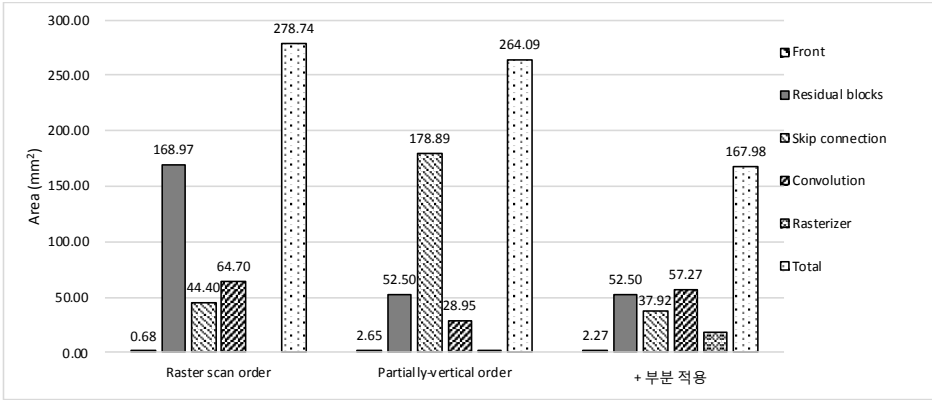


그림 3.9 SRResNet에의 부분적 수직 연산 적용 결과

하드웨어 구조를 구성할 수 있다. 표 3.4는 영상 분류용 CNN내 하나의 SRAM에 여러 채널의 $fmap$이 저장되는 경우 SRAM 면적 변화를 보인다. N^S가 2인 경우와 4인 경우에 대해 각각 SRAM의 면적이 1.06배와 1.10배 감소한다.

그림 3.9는 SISR용 CNN인 SRResNet에 부분적 수직 순서를 적용하는 방법을 보인다. 표 3.1의 CNN 구조들은 hidden 컨볼루션 레이어 사이 skip connection이 없는 구조인 반면 SRResNet은 hidden 컨볼루션 레이어 사이 skip connection과 residual block들이 존재 한다. 그림 3.9의 raster scan order는 듀얼 포트 SRAM을 사용하는 구조이다. Partially-vertical order와 부분 적용은 각각 그림 3.7의 convolution 35와 convolution 36까지 부분적 수직 연산을 적용한 결과이다. SRResNet 전체에 부분적 수직 순서 연산을 적용하는 경우 on-chip 메모리인 SRAM의 면적은 94.96%로 감소한다. 반면, 부분적 수직 연산을 convolution 35까지 적용한 경우 on-chip SRAM 면적은 60.26%로 감소한다. SRResNet의 전체에 부분적 수직 순서를 적용하는 경우 skip connection의 $fmap$을 저장하기 위한 SRAM의 면적이 4.03배 증가하

여 on-chip SRAM 면적 감소 효율을 저하한다. Residual block의 SRAM 면적은 기존의 래스터 스캔 순서와 비교하여 제안하는 방법은 31.07%로 감소한다. Convolution 35까지 부분적 수직 순서를 적용한 경우 residual block을 제외한 모든 컨볼루션 레이어의 SRAM 면적은 88.52%로 감소한다. Convolution 36은 부분적 수직 순서가 적용되지 않아 듀얼 포트 SRAM을 사용하며 convolution 34이하의 hidden 컨볼루션 레이어와 비교하여 4배 해상도의 $fmap$을 여덟 라인을 저장하며 이는 싱글 포트 SRAM을 사용하는 3×3 hidden 컨볼루션 레이어 대비 최대 30.9배의 면적을 차지한다.

제 4 장 영상의 context 보존을 위한 필터 재구성을 적용한 CNN 하드웨어 구조

본 장은 기존의 VDSR 구조를 사용하여 영상 super-resolution을 연산을 실시간으로 수행하기 위한 스트리밍 (streaming) 하드웨어 구조에 적합한 컨볼루션 레이어의 필터 구조를 재구성 하는 방법을 제안한다. SISR용 CNN의 스트리밍 하드웨어 구조는 CNN을 구성하는 모든 레이어를 파이프 라인 구조로 구성해야 한다. 이는 CNN내 컨볼루션 레이어의 중간 결과 $fmap$ 들을 모두 on-chip SRAM에 저장해야 함을 의미하며 CNN의 크기에 제약을 야기한다.

CNN의 on-chip SRAM 크기는 컨볼루션 레이어의 필터 높이에 비례 한다. 따라서 컨볼루션 레이어의 필터 높이가 작을수록 적은 on-chip SRAM이 사용 된다. 그러나, VDSR은 컨볼루션 필터의 크기가 3×3으로 대칭 구조 필터 중 최소 크기이다. 컨볼루션 필터의 높이를 1으로 변경하는 경우 세로 방향의 컨텍스트 활용이 불가능하며 이는 SISR 성능 저하를 야기한다. 본 장은 VDSR의 컨볼루션 필터의 형태를 변경하여 on-chip SRAM의 크기를 줄이기 위한 방법을 제안한다. 제안하는 필터는 1D 형태를 갖으며 VDSR의 on-chip SRAM 크기를 절반으로 감소한다. 더불어 sub-pixel 컨볼루션 레이어 [20]를 적용함으로 입력 영상 기준으로 context 크기를 유지하면서 CNN의 컨볼루션 레이어 개수를 절반으로 감소 한다. 제안하는 방법을 적용하여 구현한 CNN 하드웨어의 하드웨어 리소스 사용량과 SISR 결과 영상을 분석하여 제안하는 방법의 효과를 실험적으로 제시한다.

4.1 SRAM 감소를 위한 제안 알고리즘

4.1.1 컨볼루션 필터의 1D 재구성 방법

제안하는 방법은 기존의 2D 형태의 컨볼루션 필터를 1D 형태의 컨볼루션 필터로 재구성 하여 컨볼루션 필터의 하드웨어 리소스 사용량을 감소한다. 제안하는 컨볼루션 필터 재구성 방법은 두 개의 연속한 컨볼루션 레이어에 대해서 적용 되며, 각각의 컨볼루션 레이어의 필터를 1D 형태로 재구성 한다. VDSR의 3×3필터는 대칭 구조 필터 중 가장 작은 크기로 더 낮은 높이의 필터로 변경하는 것이 불가능 하다. 따라서 서로 연속한 컨볼루션 레이어간 병합 (layer merge) 을 적용하여 컨볼루션 필터의 크기를 증가시킨다. 그림 4.1에서 두 개의 $\mathrm{Conv}(3\times3,1,1)$이 $\mathrm{Conv}(5\times5,1,1)$으로 변경되는 과정은 컨볼루션 레이어 병합 방법을 보인다. 그림 4.1의 $\mathrm{Conv}(H\times W,N,M)$ 은 N 개의 채널로 구성된 입력 특징 맵에 대하여 $H\times W$ 크기의 필터가 M 개로 구성된 컨볼루션 레이어를 의미한다. H와 W 는 각각 컨볼루션 필터의 높이와 너비를 의미한다. 컨볼루션 레이어 병합은 서로 다른 두 개의 연속한 컨볼루션 레이어의 컨텍스트 (context; =receptive field) 크기를 유지하면서 컨볼루션 레이어의 필터 크기를 증가한다. 따라서 3×3필터 두 개는 5×5 크기의 한 개 컨볼루션 필터로 병합 된다. 병합된 컨볼루션 레이어는 병합 이전의 컨볼루션 레이어와 동일한 컨텍스트 크기를 갖으나 비선형성 (non-linearity)는 절반으로 감소한다. CNN의 비선형성은 CNN의 성능과 밀접한 연관이 있다. 따라서 비선형성을 유지 하는 것이 중요하다.

본 연구는 CNN의 비선형성을 유지하기 위해 한 개의 병합 컨볼루션 레이어를 세로 필터 컨볼루션 레이어와 가로 필터 컨볼루션

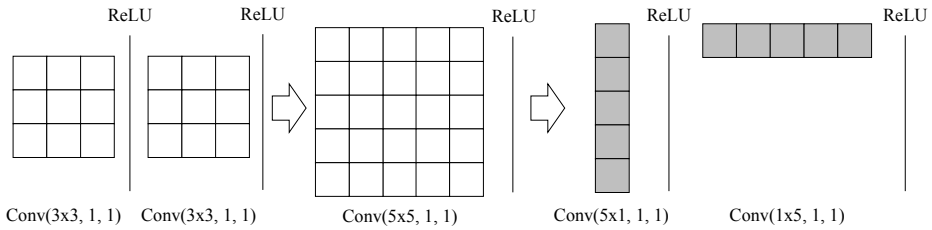


그림 4.1 컨볼루션 필터의 컨텍스트 보존 1D 재구성 방법

레이어로 재구성 한다. 병합된 2D 구조의 컨볼루션 레이어를 두 개 사용하는 경우 CNN의 병합 이전의 두 개의 컨볼루션 레이어와 비교하여 on-chip SRAM 크기가 2배 증가하는 동시 곱셈기의 수도 증가하는 문제가 있다. 따라서 on-chip SRAM의 크기를 유지하면서도 곱셈기를 줄일 수 있는 1D 필터 구조를 사용한다.

컨텍스트 보존 1D 재구성 방법은 재구성 이전의 두 개의 컨볼루션 레이어들과 비교하여서 on-chip SRAM 사용량이 동일하나 세로 필터의 크기가 5이므로 더 작은 대칭 구조의 컨볼루션 필터로 변경하는 것이 가능하다. 더불어 재구성 이전의 두 개의 컨볼루션 레이어들과 비교하여서 곱셈기 감소 효과를 보인다. 첫 번째와 마지막 컨볼루션 레이어를 제외한 나머지 컨볼루션 레이어 내 컨볼루션 필터의 곱셈기 개수를 비교한 결과는 식 (4.1)과 같다.

Ratio of the number of multipliers

$$= \frac{((2R_H - 1) \times M \times N) + \left((2R_W - 1) \times M \times N\right)}{2 \times (R_H \times R_W \times M \times N)} = \frac{(R_H + R_W - 1)}{(R_W \times R_H)} \qquad (4.1)$$

VDSR의 연속한 두 컨볼루션 레이어인 $i-1$, i 번째 컨볼루션 레이어의 곱셈기 수는 $\left(R_H \times R_W \times M \times N(i-1)\right) + \left(R_H \times R_W \times M \times\right.$

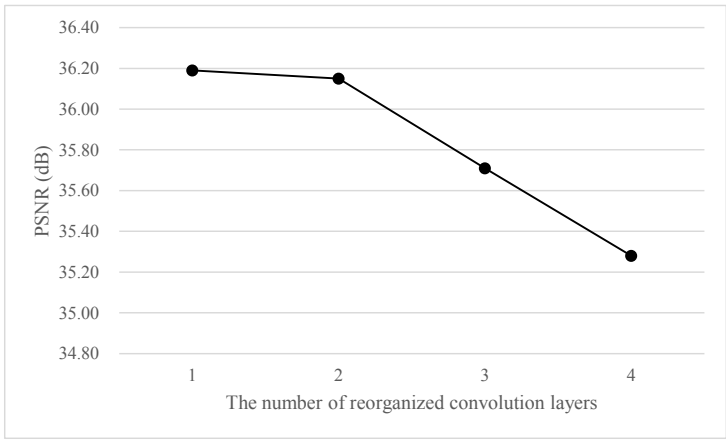


그림 4.2 필터 1D 재구성 컨볼루션 레이어 개수 대비 영상 SISR 성능 비교

$N(i)\big)$ 이며, VDSR에 컨볼루션 1D 필터 재구성을 적용한 경우의 곱셈기 수는 $\big((2R_H-1)\times M\times N(i-1)\big)+\big((2R_W-1)\times M\times N(i)\big)$ 이다. VDSR의 모든 숨겨진 컨볼루션 레이어들 (hidden convolution layers) 에 대해서 R_H 및 R_W 는 3이며 M 과 N 은 64이다. 따라서, 컨볼루션 1D 필터 재구성 방법은 1D 필터 재구성 이전의 컨볼루션 레이어와 동일한 크기의 SRAM을 사용 하면서도 VDSR의 곱셈기를 55.6%로 감소한다.

제안하는 방법에서 컨볼루션 필터 1D 재구성 방법은 비선형성 정도를 유지하도록 연속하는 두 개의 컨볼루션 레이어에 적용한다. 이 재구성 방법을 연속하는 세 개 또는 네 개의 컨볼루션 레이어에 적용하는 경우 CNN을 구성하는 곱셈기의 개수를 추가적으로 감소할 수 있으나, CNN의 비선형성이 감소하고 이는 영상 super-resolution의 성능 저하를 야기한다. 그림 4.2는 컨볼루션 필터 1D 재구성이 적용된 연속한 컨볼루션 레이어의 개수에 따른 Set 5 영상들에 대한 SISR 결과 영상의 PSNR을 비교한 결과를 보인다. 재구성된 컨볼루션 레이어의

개수가 1인 경우는 2D 필터 구조를 그대로 사용하는 경우를 의미한다. 이 2D 필터 구조를 사용한 CNN은 아홉 개의 컨볼루션 레이어 ($M = 12$)와 한 개의 sub-pixel 컨볼루션 레이어 ($M = 4, s = 2$)로 구성 된다. 두 개의 연속한 컨볼루션 레이어에 대해 필터 1D 재구성 방법 적용시 PSNR 저하가 0.04 dB로 매우 작은 반면 세 개의 컨볼루션 레이어와 네 개의 컨볼루션 레이어에 대해서 필터 1D 재구성 방법 적용시 각각 PSNR 저하가 0.48 dB 와 0.91 dB로 매우 크다.

4.1.2 CNN 라인 버퍼 감소 방법

컨볼루션 필터 1D 재구성 방법이 VDSR에 적용된 경우, 수평 방향 필터 (horizontal filter)는 세로 방향의 $ifmap$내 세로 방향 특징 정보를 사용하지 않으므로 라인 버퍼를 사용하지 않는다. 반면에, 세로 방향 필터 (vertical filter)는 $ifmap$ 내 가로 방향의 특징 정보를 사용하지 않으나 세로 방향의 특징 정보는 사용한다. 필터 1D 재구성 방법은 2D 형태의 필터를 1D 형태로 변경함으로 하드웨어 리소스 사용량을 감소하였으나, 세로 방향의 필터 크기가 증가하므로 라인 버퍼 사용량은 이 재구성 방법을 사용하기 전과 동일하다. 각각의 컨볼루션 레이어는 $(R_H - 1) \times N$개의 라인 버퍼를 사용하며 세로 방향의 필터 크기인 R_H를 감소함으로 라인 버퍼 사용량을 감소할 수 있다. 그림 4.3는 컨볼루션 필터 1D 재구성이 적용된 컨볼루션 레이어들의 라인 버퍼 감소 기법 방법을 보인다. 5×1필터는 4 라인의 $fmap$ 을 on-chip SRAM에 저장한다. 1×5 필터는 on-chip SRAM을 사용하지 않는다. 하드웨어 리소스 사용량과 비례하는 곱셈기 수를 유지하면서 라인 버퍼의 수를 감소하기 위해서는 수직 방향 필터의 크기를 감소한다. 이와 동시에

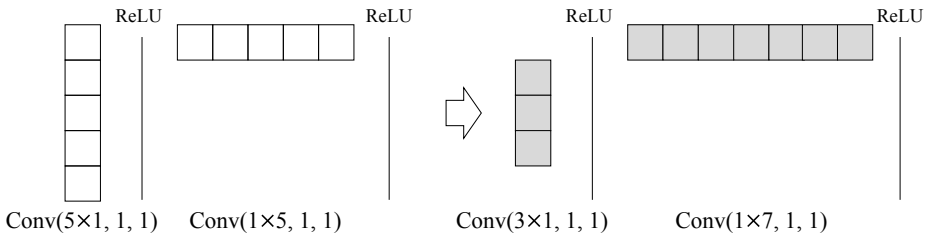


그림 4.3 CNN 라인 버퍼 감소를 위한 컨볼루션 필터 구조

SISR 결과 영상의 화질 열화를 최소화 하기 위하여 가로 방향 필터의 크기를 증가하는 방법으로 CNN의 라인 버퍼 수를 감소 할 수 있다.

제안하는 방법은 그림 4.3과 같이 두 개의 5×1, 1×5 필터 들을 3×1, 1×7 필터로 변경한다. 5×1, 1×5 필터 들을 라인 버퍼 감소 방법을 적용하여 대칭형 필터 구조로 재구성 하는 경우 1×1, 1×9 필터 들로 구성 하는 것이 가능하다. 이와 같은 구성은 라인 버퍼를 사용하지 않으나 세로 방향의 context가 없어 3.1.1장의 10개 컨볼루션 레이어로 구성된 CNN에 적용시 3×1, 1×7 형태 필터와 비교하여 2배 확대 비율 결과 PSNR이 3.29 dB 하락하여 영상 화질에 큰 손실을 야기한다. 더불어 5×1, 1×5 필터 들을 3×1, 1×7 필터로 변경하는 경우 super-resolution 성능을 크게 하락하지 않는 동시에 라인 버퍼의 개수를 절반으로 감소할 수 있다.

4.1.3 Sub-pixel 컨볼루션 레이어 적용

Sub-pixel 컨볼루션 레이어는 SRCNN 알고리즘의 동작 속도를 향상 하기 위해 제안 된다. 기존의 SISR용 CNN들은 저해상도 영상을 확대 비율에 따라서 Bicubic 보간법을 적용하여 미리 확대하여 CNN의

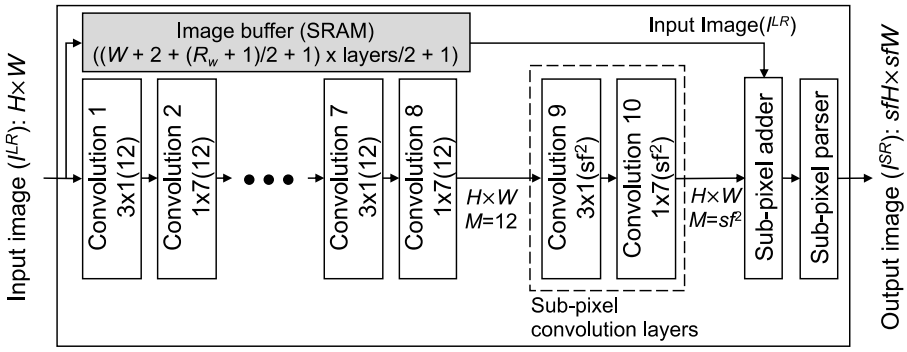


그림 4.4 제안하는 SISR을 위한 CNN 하드웨어 구조

입력 영상과 출력 영상의 해상도는 동일하다. 그러나 sub-pixel 컨볼루션 레이어를 적용하는 경우 super-resolution image (I^{SR})의 해상도는 마지막 컨볼루션 레이어에서 결정 된다. 이는 CNN의 모든 컨볼루션 레이어가 저해상도 영상을 기준으로 연산이 수행됨을 의미하며, 하드웨어의 기준으로는 마지막 컨볼루션 레이어의 필터 개수 M 에 따라서 I^{SR} 의 확대 비율이 결정된다. 따라서 SISR 확대 비율에 무관하게 컨볼루션 레이어의 라인 버퍼 크기가 동일하다.

4.2 SISR용 CNN 하드웨어 구조

4.2.1 SISR을 위한 하드웨어 구조

본 장은 앞서 제안한 두 가지 필터 구조 변경 방법과 sub-pixel 컨볼루션 레이어를 적용한 CNN의 하드웨어 구조를 제안한다 [20]. 이 CNN의 하드웨어 구조는 디스플레이 장치에 적용하기 위하여 픽셀 단위 실시간 동작이 요구되어 스트리밍 구조를 갖으며 입력 영상은 래스터 스캔 순서 (raster scan order) 로 입력되어 동일한 순서로 I^{SR}이 출력된다. 그림 4.4은 제안하는 SISR을 위한 CNN의 하드웨어 구조를 보인다.

제안하는 SISR용 CNN은 영상과 컨볼루션 레이어의 연산 중간 결과인 $fmap$을 저장하기 위한 SRAM으로 구성된 버퍼와 열 개의 컨볼루션 레이어 들로 구성 된다. 각각의 컨볼루션 레이어는 $R_H \times 1$ 또는 $1 \times R_W$ 의 컨볼루션 연산을 수행한다. 컨볼루션 레이어 1부터 컨볼루션 레이어 8까지는 12개의 채널의 $ofmap$ 을 출력 하며 컨볼루션 레이어 9와 10은 확대 비율의 제곱 개수의 채널을 갖는 $ofmap$을 출력 한다. 이 두 개의 컨볼루션 레이어는 컨볼루션 결과에 활성화 함수를 적용하지 않는다. Sub-pixel adder 모듈은 10 번째 컨볼루션 레이어에서 출력된 $ofmap$의 특징에 Image buffer에 저장되어 있는 입력 영상을 더한다. 제안하는 CNN 하드웨어 구조는 Y 채널에 대해 동작하므로 입력 영상의 크기는 $H \times W \times 1$ 이다. 10 번째 컨볼루션 레이어의 $ofmap$의 크기는 $H \times W \times s^2$이며 sub-pixel adder 모듈은 이 $fmap$의 2차원 좌표와 동일한 위치의 입력 영상을 $fmap$의 모든 채널에 더한다. Sub-pixel parser는 식 (2.1)에 따라서 sub-pixel adder 모듈의 출력 데이터를 배치한다.

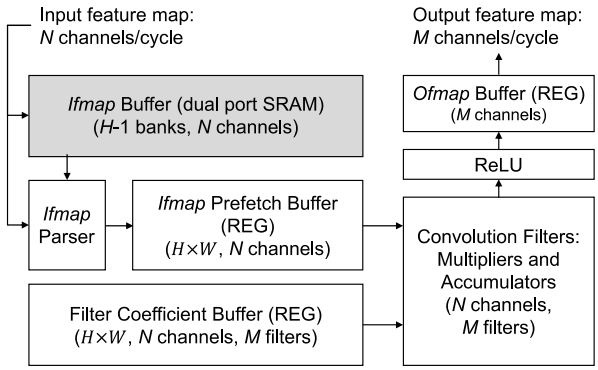


그림 4.5 제안하는 컨볼루션 레이어 하드웨어 구조 (dual port SRAM)

4.2.2 컨볼루션 레이어 하드웨어 구조

스트리밍 구조의 CNN을 구성하는 컨볼루션 레이어는 모든 입력 특징 맵의 N개의 채널에 대해서 M개의 필터 연산을 병렬로 수행한다. 그림 4.5는 CNN 하드웨어를 구성하는 컨볼루션 레이어 모듈의 구조를 보인다. 컨볼루션 레이어는 이전 컨볼루션 레이어의 연산 결과 $ofmap$을 저장하기 위한 SRAM으로 구성된 라인 버퍼인 $ifmap$ buffer와 컨볼루션 연산을 하기 위한 로직으로 나뉘어진다. $Ifmap$ buffer는 래스터 스캔 순서로 입력되는 $ifmap$을 저장하는 동시에 동일한 순서로 컨볼루션 연산 결과를 출력하기 위해 읽기 및 쓰기 동작을 동시에 수행할 수 있는 듀얼 포트 SRAM (dual port SRAM)을 사용한다. 3×3 필터 연산 시 라인 버퍼에 저장되어 있는 두 라인의 $ifmap$과 새로 입력되는 $ifmap$으로 필터 연산이 가능하다. 따라서 *ifmap* buffer는 $R_H - 1$개의 뱅크 (bank) 들로 구성 되어 있으며, 하나의 뱅크에는 N 개 채널의 $ifmap$을 저장 한다. $Ifmap$ parser는 래스터 스캔 순서로 입력되는 $ifmap$ 데이터와 라인 버퍼에 미리 저장된 $ifmap$을 각

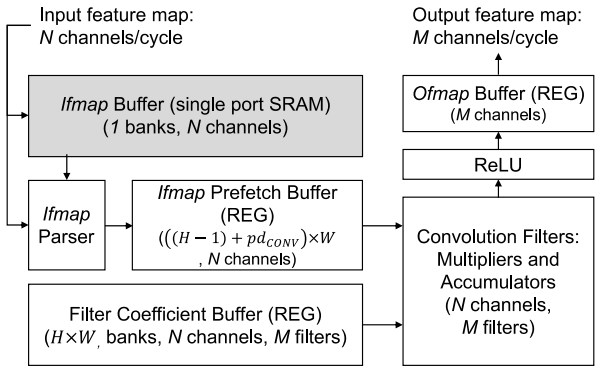


그림 4.6 부분적 수직 순서 적용 컨볼루션 레이어 하드웨어 구조 (single port SRAM)

채널마다 $R_H \times R_W$ 형태로 변경하여 컨볼루션 연산 전 $ifmap$ prefetch buffer 모듈에 저장한다. $Ifmap$ prefetch buffer 모듈에 저장된 입력 특징들과 filter coefficient buffer에 미리 저장되어 있는 필터 계수를 convolution filters 모듈에 입력하여 컨볼루션 연산을 수행하며, 컨볼루션 연산 결과는 활성화 함수인 ReLU를 거쳐 $ofmap$ $buffer$에 저장 된다. 컨볼루션 레이어 모듈은 매 클락 사이클마다 N 개의 특징 데이터를 입력 받아 M개의 특징 데이터를 출력 한다.

4.2.3 부분적 수직 순서 적용 컨볼루션 레이어 하드웨어 구조

필터 구조가 변경된 CNN 하드웨어 구조는 제 3장의 부분적 수직 순서 컨볼루션 연산이 적용 가능하다. 부분적 수직 순서를 모든 컨볼루션 레이어에 적용하는 경우 제 4.2.2장의 컨볼루션 레이어 하드웨어 구조에 사용되는 듀얼 포트 SRAM을 싱글 포트 SRAM으로 변환하여 on-chip 메모리 구성이 가능하며, 이는 듀얼 포트 SRAM을

기준으로 절반 크기의 SRAM을 사용 하는 것과 동일하다. 그림 4.6은 부분적 수직 순서를 CNN 하드웨어에 적용하는 경우의 컨볼루션 레이어 하드웨어 구조를 보인다.

부분적 수직 순서를 적용한 컨볼루션 레이어는 $ifmap$ buffer와 $ifmap$ prefetch register의 구조가 변경 된다. $ifmap$은 이전 컨볼루션 레이어로부터 부분적 수직 순서로 입력 되며 $ifmap$ buffer에 저장 된다. $ifmap$ buffer는 두 라인의 $fmap$ 을 하나의 싱글 포트 SRAM에 저장한다. 부분적 수직 순서 연산을 위해 $ifmap$ prefetch register는 세로 방향의 크기가 $pd_{CONV}-1$ 만큼 증가하여 듀얼 포트 SRAM 구조를 사용하는 경우와 비교하여 $(pd_{CONV}-1)\times W\times N$ 만큼의 레지스터를 추가로 사용한다. 이외의 다른 모듈의 구조는 래스터 스캔 순서로 동작하는 컨볼루션 레이어와 동일하다. 컨볼루션 연산 결과 feature는 $ofmap$ register를 거쳐 다음 컨볼루션 레이어로 전달 된다.

4.2.4 영상 super-resolution을 위한 CNN 하드웨어 구조의 구성

SISR용 하드웨어를 구성하기 위해 SISR결과 영상의 화질과 하드웨어 리소스 사용량 간 균형 (tradeoff)가 필요하다. 본 장은 이 균형을 위해 CNN의 크기를 줄이는 방법에 대한 내용은 다음과 같다.

제안하는 CNN 하드웨어는 sub-pixel 컨볼루션 레이어의 효과로 총 10개의 컨볼루션 레이어를 사용 한다. VDSR의 콘텍스트 크기는 41×41이다. 이 콘텍스트는 저해상도 입력 영상에 Bicubic 보간법을 적용한 영상에 대한 콘텍스트 크기로 Bicubic 보간법을 적용 하기 이전의 저해상도 영상에 대한 콘텍스트 크기는 $\left(\frac{41-1}{s}+1\right)^2$이다. 따라서 확대 비율이 2인 경우 저해상도 입력 영상에 대한 콘텍스트 크기는

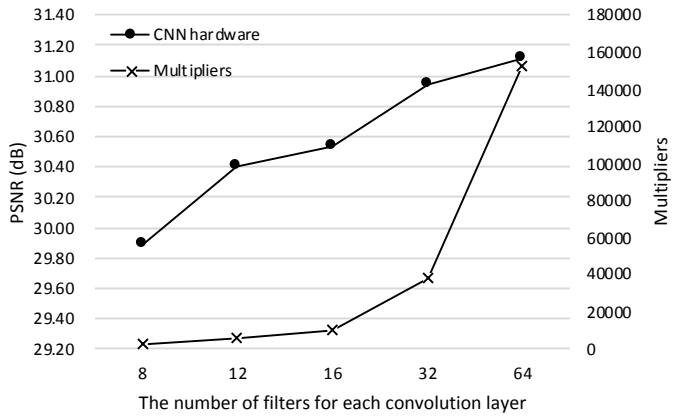


그림 4.7 CNN 하드웨어의 필터 개수에 따른 SISR 성능

21 × 21이다. 이는 VDSR에 sub-pixel 컨볼루션 레이어를 적용시 10개의 컨볼루션 레이어를 사용하는 경우의 컨텍스트 크기와 동일하다. 이는 CNN의 SRAM 사용량과 하드웨어 로직 리소스 사용량을 절반으로 감소한다.

컨볼루션 레이어의 개수를 감소하는 방법과 더불어 CNN의 컨볼루션 필터 개수를 감소하는 방법을 적용하는 경우 CNN의 SRAM 사용량과 하드웨어 로직을 추가적으로 감소할 수 있다. VDSR을 기반으로 하는 제안하는 CNN 하드웨어는 sub-pixel 컨볼루션 레이어를 제외한 모든 컨볼루션 레이어가 동일한 개수의 필터를 갖는다. 따라서 CNN의 필터 개수를 감소하는 경우 필터에 사용되는 곱셈기의 개수는 필터 개수의 제곱에 비례하여 감소한다. 그림 4.7는 확대 비율 2를 적용시 각각의 CNN 구조에 대해서 필터 개수에 따른 SISR 성능을 비교한다. 성능을 비교하기 위해 PSNR을 측정하기 위한 영상은 'Set5', 'Set14', 'BSD100', 'Urban100' 영상들을 사용한다. CNN hardware는 VDSR에 필터 1D 재구성 기법, 라인 버퍼 감소 기법 및 sub-pixel 컨볼루션 레이어를 적용한 구조이다. 위의 실험 결과에서 제안하는

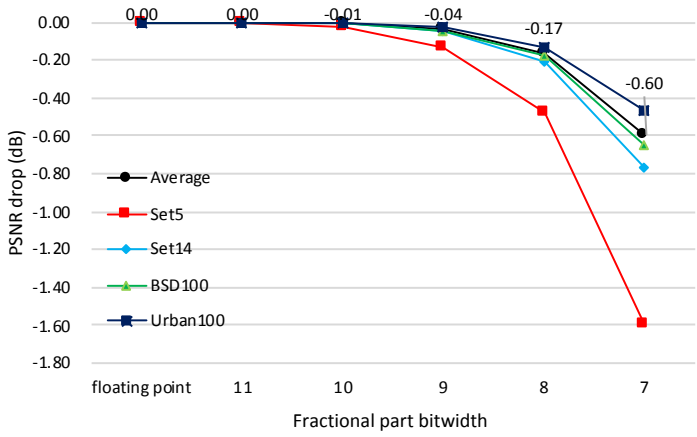


그림 4.8 CNN 하드웨어의 bit precision에 따른 SISR 성능

CNN 구조는 M=64을 기준으로 PSNR이 0.1 dB 이내로 유지 하는 최소 M인 12를 선택 한다. CNN hardware의 $M = 64$인 경우 31.11 dB 이며, $M = 12$인 경우 30.40 dB 이다.

CNN를 소프트웨어로 동작하는 경우 32-bit 부동 소수점 기반의 연산이 주로 활용된다. 그러나 부동 소수점 기반의 연산은 많은 하드웨어 리소스를 요구하기 때문에 하드웨어 구현에 부적합 하며 고정 소수점을 주로 사용한다. 그림 4.8 는 고정 소수점을 사용하는 경우 소수 부분의 비트 개수에 따른 SISR성능을 비교한 결과를 보인다. SISR 성능을 비교하기 위해 M을 선택하는 조건과 동일하게 확대 비율 2로 'Set5', 'Set14', 'BSD100', 'Urban100' 영상들을 확대하는 경우에 대해 PSNR을 측정 한다. 소수 부분에 11 bits를 사용하는 경우 평균 PSNR 저하가 발생하지 않으므로 제안하는 하드웨어 구조는 모든 데이터의 소수 부분을 11 bits로 표현 한다. 특징 맵의 정수 부분은 3 bits를 사용하며 필터 계수는 음수를 표현하기 위해 정수 부분을 4 bits로 표현 한다.

4.3 실험 결과

본 장은 제안하는 SISR용 CNN 하드웨어의 실험 결과를 제시한다. 실험 결과를 토대로 제안하는 방법의 효율성을 평가한다.

4.3.1 CNN 하드웨어 합성 결과

제안하는 SISR용 CNN 하드웨어는 Verilog HDL을 사용하여 구현한다. 이 CNN 하드웨어는 시뮬레이션 및 ASIC 합성 결과를 토대로 검증 된다. ASIC 합성은 65−nm 공정으로 50MHz 동작 주파수로 한다. 제안하는 CNN 하드웨어 구조 중 Image buffer 모듈 및 각 컨볼루션 레이어 내 $ifmap$ buffer 는 SRAM으로 분류되며 이외의 모든 모듈들은

표 4.1 제안하는 CNN 하드웨어의 리소스 사용량 결과. 입력 영상의 해상도: 256×256 (SRAM: word, Logic: K NAND2)

		Convolution layer + ReLU					Image Buffer	Etc.	Total
Layer		1	2, 4, 6, 8	3, 5, 7	9	10			
SRAM		512	0	6144	6144	0	1316	0	26404
Logic	$sf = 2$	56.6	1456.7	626.4	212.9	165.9	0	2.1	8143.3
	$sf = 3$	56.6	1456.7	626.3	471.1	822.8	0	4.6	9060.6
	$sf = 4$	56.6	1456.7	626.4	832.8	2585.0		8.2	11188.3

표 4.2 확대 비율 2인 CNN들의 하드웨어 사용량 비교 (곱셈기/SRAM(word))

Method	Layers	Convolution layer	Image Buffer	Total
VDSR	20	664704/1246208	0/10301	664704/1256509
Low-rank approximation	20	455424/1246208	0/10301	455424/1246208
Context-preserving 1D filter	20	369280/1181696	0/10301	369280/1191997
+ Vertical filter reduction	20	369280/590848	0/10301	369280/601149
+ Sub-pixel convolution layer	10	152624/147712	0/1316	152624/149028
Proposed	10	5620/25088	0/1316	5620/26404

로직으로 분류 된다. 표 4.1은 제안하는 CNN 하드웨어의 리소스 사용량을 보인다. 확대 비율 2에 대해서 제안하는 CNN 하드웨어는 8143.3K의 NAND2 게이트를 사용한다. 확대 비율이 2나 3으로 증가하는 경우 sub-pixel 컨볼루션 레이어인 아홉 번째 및 열 번째 컨볼루션 레이어의 로직 크기가 증가한다. 홀수 번째 컨볼루션 레이어들은 3×1 크기의 필터로 구성되므로 $fmap$을 저장하기 위한 라인 버퍼를 사용하며 Image buffer와 합산하여 26,404개의 feature 들을 저장한다. 표 4.2는 제안하는 CNN하드웨어 구조의 하드웨어 리소스 사용량을 비교한다. SRAM 사용량을 비교하기 위해 입력 영상의 크기는 256×256로 정한다. VDSR의 스트리밍 구조의 하드웨어 구현 연구 결과가 없으므로 CNN에 필요한 곱셈기의 개수를 비교하며, 기존의 super-resolution 용 CNN에 필요한 Bicubic 보간법을 연산하는 모듈은 하드웨어 리소스 사용량에서 제외 한다. 표 4.2의 Low-rank approximation은 [29] 의 'Scheme2'를 VDSR에 적용한 방법으로 한 개의 3×3 컨볼루션 레이어를 두 개의 컨볼루션 레이어들로 나누며 각각의 컨볼루션 레이어는1×3, 3×1 크기의 필터를 갖는다. Low-rank approximation은 SPVDSR의 첫 번째 컨볼루션 레이어를 제외한 나머지 컨볼루션 레이어들에 적용 된다. VDSR은 664,704개의 곱셈기를 사용하는 반면 제안하는 하드웨어는 5,620개의 곱셈기를 사용한다. 이는 VDSR보다 118.27배 로직을 적게 사용하는 것을 의미한다. 제안하는 하드웨어 구조에 Full-HD 영상을 입력하여 확대 비율 2를 적용하여 UHD 해상도를 출력하는 경우 333.1kB의 SRAM이 필요 하다.

표 4.3은 제안하는 하드웨어 구조에 부분적 수직 방향 연산 방법을 적용한 경우 SRAM의 면적을 보인다. SRAM 면적을 측정하기 위해 3장과 동일하게 CACTI 시뮬레이터를 사용하며, super-resolution 영상의 해상도는 Full HD이다. 따라서, CNN은 960×540 해상도의 $fmap$ 들을 on-chip SRAM에 저장한다. 부분적 수직 순서를 적용한 경우 CNN의 첫 번째 컨볼루션 레이어의 멀티 뱅크 SRAM 구조와 CNN 연산 결과의 부분적 수직 순서의 $ofmap$ 스트림을 래스터 스캔 순서로 변경하기 위한 rasterizer 모듈로 인해 SRAM의 용량은 5.7 kB 증가한다. 그러나, 부분적 수직 순서를 적용 하는 경우 기존의 듀얼 포트 SRAM을 싱글 포트 SRAM으로 변경하며 CNN의 SRAM 면적을 64.52%로 감소한다. 이 중 hidden 컨볼루션 레이어의 면적은 45.40%로 감소한다. Front 컨볼루션 레이어와 rasterizer 모듈의 SRAM 면적은 듀얼 포트 SRAM을 사용하는 CNN 대비 3.52배 증가하나 이 모듈들은 한 개의 채널을 갖는 데이터를 저장하며 전체 SRAM 면적의 34.01%로 hidden 컨볼루션 레이어의 SRAM 면적과 비교하여 적은 면적을 차지한다. 부분적 수직 방향 연산은 $ifmap$ prefetch register의 크기를 증가하며 듀얼 포트 SRAM과 비교하여 면적이 3.42배 증가하나 이는 부분적 수직 방향 연산을 적용한 후 SRAM의 면적과 비교하여 2.65%로 매우 작은 면적을 차지하므로

표 4.3 부분적 수직 순서 적용 CNN 하드웨어 SRAM 면적 비교

CNN 구조	M, layers	SRAM (kB)	SRAM area (mm^2)				Register area (mm^2)	Regiter area/ SRAM area (%)
			Front	Hidden	Rasterizer, Identity	Total		
Proposed	12, 10	193.2	0.056	2.709	0.124	2.889	0.014	0.50
+ 부분적 수직 순서		221.4	0.233	1.230	0.401	1.864	0.031	2.65

무시할 수 있다.

4.3.2 학습 방법

제안하는 CNN 하드웨어 구조는 VDSR의 학습과 동일하게 291장의 영상을 사용 한다 [30]. 291장의 영상에는 반전 및 회전을 적용하여 데이터를 증원 (augmentation) 한다. CNN에 입력되는 영상은 21×21 크기의 패치를 사용하며 학습을 위한 정답 영상은 입력 영상의 가로와 세로에 확대 비율만큼 곱하여 생성한다. VDSR은 확대 비율 2, 3 및 4를 함께 학습 하지만 제안하는 CNN 하드웨어는 입력 영상과 출력 영상의 해상도가 다르므로 각각의 확대 비율에 대해 학습을 수행한다. 제안하는 CNN 하드웨어는 adaptive moment estimation (ADAM) optimizer로 80 에폭 (epoch) 동안 학습 하여 L2 손실을 감소한다. CNN의 learning rate는 0.001이다. Low-approximation은 100에폭 동안 stochastic gradient descent optimizer를 사용하여 L2 손실을 감소하도록 학습 하였으며 초기 학습률은 0.1이며 40 에폭마다 학습률을 10배씩 감소 한다.

4.3.3 Super-resolution 결과

제안하는 CNN 하드웨어의 super-resolution 성능은 PSNR 및 SSIM을 통하여 객관적으로 비교 한다. 테스트 영상은 'Set5' [31], 'Set14' [31], 'B100' [32] 및 'Urban100' [8]영상들을 사용 한다. 표 4.4은 제안 하는 방법을 포함하여 VDSR, Low-rank approximation 등의 기존 방법들의 super-resolution 결과를 보인다.

컨텍스트 보존 1D 필터 구조는 on-chip SRAM의 크기를 유지하며

곱셈기를 감소하는 효과가 있으므로 기존 연구의 low-rank approximation과 비교할 수 있다. Low-rank approximation방법의 'Scheme 2'는 한 개의 3×3 필터를 3×1과 1×3 필터로 분해하는 방법으로 곱셈기의 수를 66.7%로 감소한다. Low-rank approximation 방법을 VDSR에 적용하는 경우 곱셈기 수를 68.52%로 감소하는 동시 VDSR대비 평균 0.04 dB 하락이 발생한다. 컨텍스트 보존 1D 필터를 VDSR에 적용하는 경우 곱셈기의 수를 55.56%로 감소하며 VDSR과 비교하여 PSNR 저하가 0.03 dB로 low-rank approximation을

표 4.4 제안하는 하드웨어 구조의 SISR 결과 (PSNR (dB)/SSIM)

		VDSR		Low-rank approximation		Context-preserving 1D filter		+ Vertical filter size reduction	
sf	Test sets	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
2	Set5	37.24	0.9583	37.16	0.9580	37.18	0.9580	37.15	0.9579
	Set14	32.80	0.9117	32.88	0.9115	32.85	0.9113	32.87	0.9117
	B100	31.73	0.8948	31.71	0.8943	31.74	0.8947	31.72	0.8948
	Urban100	30.35	0.9040	30.28	0.9082	30.33	0.9095	30.31	0.9094
	Average	31.29	0.9015	31.26	0.9032	31.29	0.9040	31.27	0.9040
3	Set5	33.37	0.9191	33.30	0.9187	33.34	0.9194	33.36	0.9192
	Set14	29.67	0.8314	29.72	0.8307	29.69	0.8311	29.75	0.8312
	B100	28.72	0.7963	28.69	0.7955	28.69	0.7966	28.71	0.7965
	Urban100	26.84	0.8200	26.75	0.8178	26.81	0.8215	26.83	0.8204
	Average	28.03	0.8122	27.98	0.8107	28.00	0.8130	28.02	0.8124
4	Set5	31.09	0.8799	31.03	0.8788	30.94	0.8790	30.98	0.8780
	Set14	27.87	0.7663	27.91	0.7652	27.85	0.7659	27.92	0.7651
	B100	27.19	0.7238	27.16	0.7230	27.15	0.7240	27.16	0.7227
	Urban100	24.96	0.7393	24.88	0.7411	24.87	0.7443	24.91	0.7429
	Average	26.30	0.7372	26.26	0.7375	26.24	0.7395	26.27	0.7382

		J.W. Chang *et al.*		M.C. Yang *et al.*		Proposed hardware			
						M = 64		M = 12	
sf	Test sets	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
2	Set5	36.20	-	33.83	-	37.11	0.9573	36.36	0.9528
	Set14	32.03	-	29.77	-	32.81	0.9109	32.32	0.9060
	B100	31.01	-	-	-	31.65	0.8940	31.22	0.8882
	Urban100	-	-	-	-	30.03	0.9050	29.02	0.8893
	Average	31.35			-	31.11	0.9015	30.40	0.8913
3	Set5	32.45	-	-	-	33.07	0.9133	32.37	0.9025
	Set14	29.03	-	-	-	29.65	0.8270	29.09	0.8176
	B100	28.20	-	-	-	28.58	0.7934	28.25	0.7849
	Urban100	-	-	-	-	26.43	0.8084	25.79	0.7868
	Average	-	-	-	-	27.76	0.8051	27.27	0.7905
4	Set5	30.09	-	-	-	30.75	0.8680	30.00	0.8477
	Set14	27.22	-	-	-	27.76	0.7592	27.24	0.7448
	B100	26.71	-	-	-	27.05	0.7198	26.73	0.7089
	Urban100	-	-	-	-	24.64	0.7300	24.08	0.7030
	Average	-	-	-	-	26.08	0.7300	25.63	0.7117

VDSR에 적용하는 것과 비교하여 컨텍스트 보존 1D 필터가 효율적임을 알 수 있다. VDSR에 컨텍스트 보존 1D 필터와 수직 방향 필터 크기 감소 기법을 적용한 경우 VDSR과 비교하여 0.02 dB 성능 저하가 발생한다. 이는 영상의 수직 방향 대신 수평 방향의 컨텍스트가 더 중요함을 의미하며 on-chip SRAM을 절반으로 감소하여도 성능 하락이 발생하지 않음을 의미한다.

표 4.5은 제안하는 CNN하드웨어와 기존의 SISR 하드웨어 구조를 비교한 결과를 보인다. T. Manabe *et al.*의 구조는 133 MHz의 clock 주파수를 사용하여 Full HD 해상도 영상을 60 fps로 출력 한다 [22]. M.C. Yang *et al.* 의 구조는 136 MHz로 동작하여 Full HD 해상도의 영상을 60 fps로 출력 한다 [21]. J.W. Chang *et al.*의 구조는 QHD 해상도의 영상을 141 fps로 출력 한다 [23]. 제안 구조는 50 MHz로 동작 하며 Full HD 해상도 영상을 96.45 fps로 출력한다. 세 하드웨어 구조가 136 MHz의 동일한 clock 주파수를 사용하는 환경을 가정하는 경우 제안하는 CNN 하드웨어는 262.34 fps로 동작한다. 제안하는 CNN 하드웨어는 'Set5' 및 'Set14' 영상들에 대해 M.C. Yang *et al.*의 방법보다 RGB 채널의 평균 PSNR이 각각 0.11 dB, 0.12 dB높으면서도

표 4.5 제안하는 CNN 하드웨어와 기존 SISR 하드웨어 비교 (sf=2)

	T. Manabe *et al.*	M.C. Yang *et al.*	J.W Chang *et al.*	Proposed
Frequency (MHz)	133	136	130	50
Input (pixels)	960× 540	960×540	1080×720	960×540
Output (pixels)	1920×1080	1920×1080	2160×1440	1920×1080
Frame rate (fps)	60	60	141	96.45
Normalized frame rate (fps, 136 MHz)	61.12	60	147.51	262.34
Memory (kB)	578.0	235.58	329 (QHD) 292.4 (FHD)	198.9
Gate count (K NAND2)	4729.47 (Est.)	1985.33	2524.12 (Est.)	8143.26
Gate count /Normalized frame rate	77.26	33.09	16.76	31.04

Gate counts/normalized frame rate는 2.05 낮아 효율적 임을 알 수 있다. T. Manabe *et al.*의 구조는 Gate counts/normalized frame rate가 77.26으로 제안 방법의 2.49배이다. T. Manabe는 super-resolution 성능 측정을 위해 자체 제작한 영상들을 사용하였으며 이는 Urban 100 테스트 영상과 유사하다. T. Manabe *et al.* 의 구조는 Bicubic 과 비교하여 평균 SSIM이 0.0308높으며, 제안하는 구조는 Urban 100 테스트 영상에 대해 평균 SSIM이 0.0442가 높다. 따라서 제안하는 구조가 기존의 두 super-resolution 하드웨어 구조와 비교하여 효율이 높다. J.W. Chang *et al.* 의 방법은 FSRCNN을 스트리밍 구조의 하드웨어로 구현한 결과로 매우 높은 동작 속도를 보인다. 해당 구조는 Gates counts/normalized frame rate가 16.76 으로 제안하는 구조와 비교하여 1.85배 작은 구조이다. 그러나 Set5, Set14, B100의 영상에 대해서 제안 구조보다 0.22 dB 낮은 SISR성능을 보인다.

그림 4.9은 제안하는 CNN 하드웨어 구조의 Baboon 영상에 대한 super-resolution 결과 영상을 보인다. Baboon 영상은 CNN 하드웨어 구조의 super-resolution 결과 Set 14 영상 들 중 가장 낮은 PSNR을 보이는 영싱이다. 결과 영상 내 적색 상자는 CNN 하드웨어의 super-resolution 결과 최소 PSNR 블록을 의미하며 녹색 상자는 영상 내 임의의 블록을 선택 하였다. 제안하는 하드웨어 구조는 VDSR과 비교하여 0.20 dB 낮으며, 출력 영상 또한 VDSR의 출력 영상과 시각적으로 유사하다. 그림 4.10은 Lena 영상의 super-resolution 결과 영상으로 Set 14에 대해서 VDSR과 CNN hardware 구조의 평균 PSNR 저하되는 만큼 PSNR 저하가 발생하는 영상이다. Baboon 영상과 동일하게 적색 상자는 Lena 영상 중 가장 낮은 PSNR을 출력하는 블록들 중 세밀한 정보가 있는 영역을 선택 하였으며, 녹색 상자는

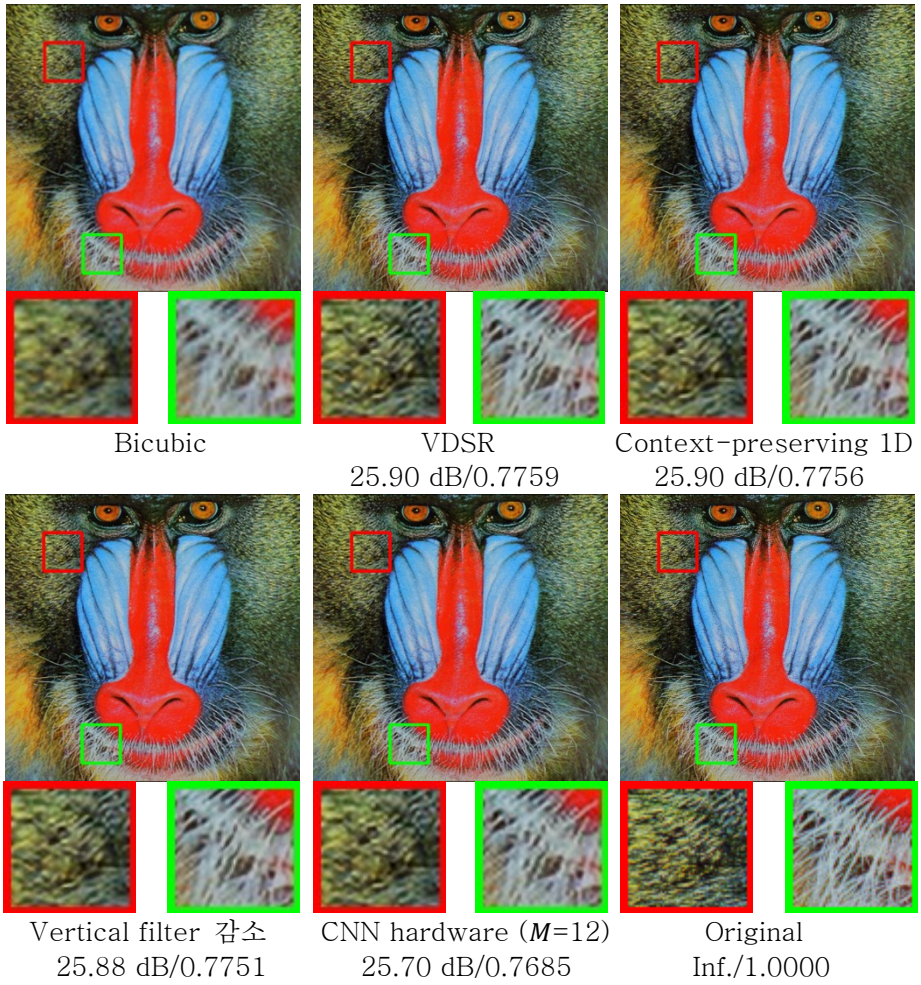


그림 4.9 Baboon (Set 14) 영상의 SISR 결과 영상 (확대 비율 2)

임의의 영역을 선택 하였다. CNN 하드웨어는 VDSR과 비교하여 해당 영상에 대해 0.41 dB 낮은 성능을 보이나 세밀한 영역에 대한 super-resolution 결과는 두 방법이 유사하다. 따라서 제안하는 CNN 하드웨어는 VDSR의 시각적 유사성을 유지하며 효율적으로 하드웨어의 크기를 줄이는 방법이 적용 됨을 확인할 수 있다.

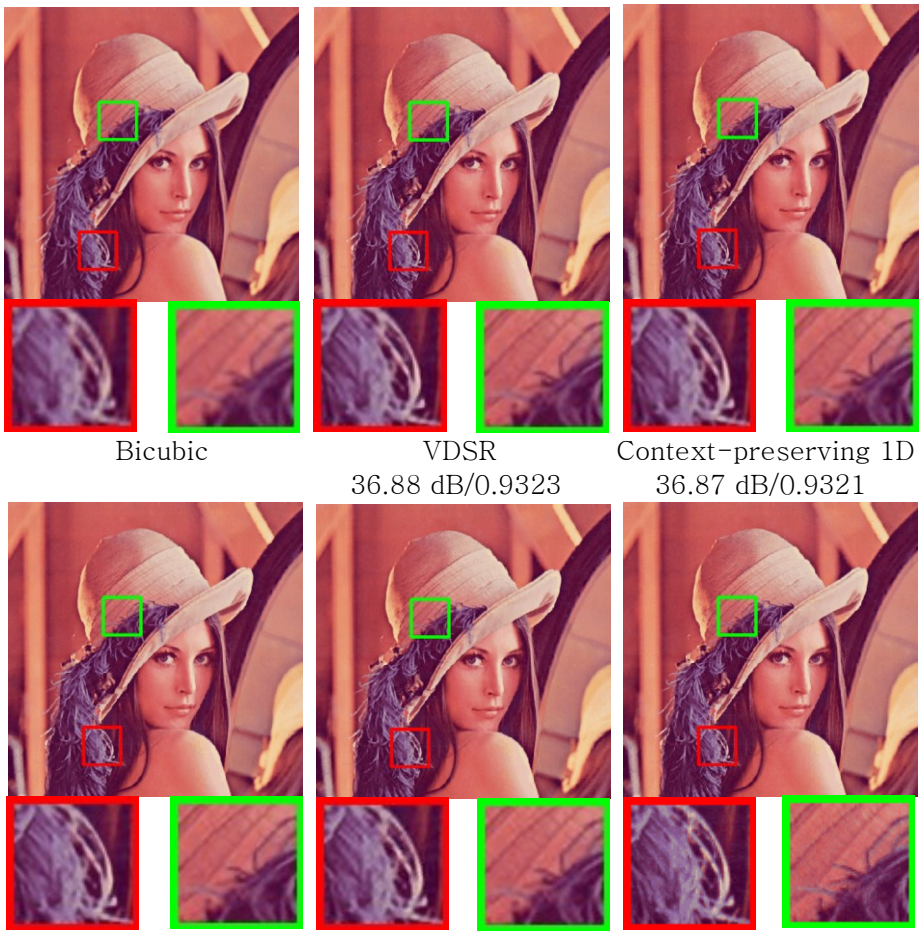


그림 4.10 Lena (Set 14) 영상의 SISR 결과 영상 (확대 비율 2)

제 5 장 SISR을 위한 해상도 보존 생산적 적대 신경망 구조

본 장은 생산적 적대적 신경망 (Generative adversarial networks; GANs)를 SISR에 적용한 SRGAN (Super-resolution generative adversarial network)를 토대로 화질을 개선하기 위한 판별 신경망 (discriminator network; D) 구조를 제안 한다 [14]. 기존의 연구들은 SISR의 성능을 개선하기 위해 생산적 신경망 (generative network; G)를 개선하는 연구에 집중 한다. 그러나 본 연구는 생산적 신경망의 구조는 유지한 채 적대 신경망의 구조를 변경 함으로 생산적 신경망이 출력하는 I^{SR}이 I^{HR}과 유사하도록 한다. 더불어 I^{SR}이 I^{HR}과 시각적으로 유사해 지기 위해 G를 학습하는 방법으로 영상의 픽셀 기반의 손실 (loss) 가 아닌 VGG (Visual geometry group) 네트워크 기반의 콘텐트 손실 (content loss)을 사용한다 [10]. 본 장은 VGG 네트워크 기반의 개선된 콘텐트 손실도 제안하여 G 가 원본 영상과 유사한 I^{SR} 을 출력하도록 학습하는 방법을 제시한다.

5.1 해상도 보존 판별 신경망 구조

SRGAN 구조에서 D는 네트워크에 입력되는 I^{HR}과 I^{SR}에 대해서 각각 진짜 영상과 가짜 영상으로 판별하는 분류 (classification) 네트워크이다. 분류 네트워크들은 영상 전체에서 포괄적인 (global) 정보를 추출하기 위해 풀링 (pooling) 레이어나 스트라이드 (s) 를

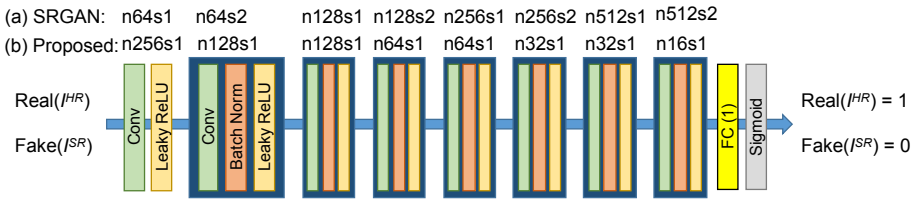


그림 5.1 SISR에 적용된 판별 신경망 구조 비교. 그림의 n과 s는 각각 $ofmap$의 채널 수와 스트라이드 크기를 의미한다. (a) 기존 SRGAN에 적용된 판별 신경망 구조; (b) 제안하는 해상도 보존 판별 신경망 구조

적용한 컨볼루션 레이어 (strided convolution layer)를 사용한다. 영상에서 포괄적인 정보만을 추출하는 것은 영상 내 물체에 대해 분류 성능의 강인함 (robustness) 를 높이는 효과가 있다. 그러나 분류와는 다르게 super-resolution 분야에서는 영상 내 지역적인 (local) 정보를 보존하는 것이 중요하다. 분류 네트워크에 사용되는 풀링 레이어나 스트라이드는 $fmap$의 해상도를 감소하며 이는 영상 내 지역적 정보를 제거하는 것과 동일하다. 따라서 본 장은 SISR을 위한 GANs 구조 사용시 지역적인 정보를 보존하는 D (resolution-preserving discriminator network; D_{RP}) 구조를 제안한다.

그림 5.1은 제안하는 D_{RP} 구조를 SISR에 적용한 예시를 보인다. 그림 5.1의 n과 s는 각각 컨볼루션 레이어에서 출력되는 $ofmap$의 채널 수와 해당 컨볼루션 레이어의 스트라이드 크기이다. 그림 5.1(a)는 기존 SRGAN [14] 에 적용 된 D의 구조이며, 그림 5.1(b)는 제안하는 D_{RP}의 구조를 보인다. D 및 D_{RP}를 구성하는 컨볼루션 레이어의 필터 크기는 모두 3×3이다. 기존의 D는 스트라이드 크기가 2인 컨볼루션 레이어가 네 개가 있다. 이는 입력 영상이 스트라이드 크기가 2인 컨볼루션 레이어를 통과 할 때마다 해상도가 $\frac{1}{s^2}$로 감소함을 의미한다. 따라서 D는

입력 영상보다 해상도가 $\frac{1}{s^{2\times4}}$ 배 작은 $fmap$ 으로부터 I^{HR} 과 I^{SR} 을 구분한다. 반면에, 제안하는 D_{RP}는 입력 영상이 네트워크를 통과 하면서 해상도 감소가 없다. 이는 네트워크가 I^{HR} 과 I^{SR} 을 구분하는데 영상의 세밀한 정보를 모두 활용 할 수 있음을 의미한다. D_{RP}를 구성하기 위한 방법은 다음과 같다. 첫 번째로 컨볼루션 레이어의 스트라이드 적용 대신 스트라이드의 크기 비율 만큼 컨볼루션 레이어의 출력 채널 수를 감소한다. 그림 5.1(a)에서 스트라이드가 적용된 컨볼루션 레이어의 바로 다음 컨볼루션 레이어인 i 번째 컨볼루션 레이어는 파라미터의 개수가 $L_{i-1}\times K_{i-1}\times\frac{1}{s^2}\times n_{i-1}\times n_{i-1}\times s$ 이다. 그림 5.2(b)에서 채널 개수를 감소한 컨볼루션 레이어의 파라미터 수 또한 $L_{i-1}\times K_{i-1}\times\frac{1}{s}\times n_{i-1}\times n_{i-1}$이다. 스트라이드가 적용된 컨볼루션 레이어나 풀링 레이어는 파라미터 수를 감소함으로 뉴럴 네트워크의 최적화를 용이하게 한다. 제안하는 컨볼루션 레이어의 채널 감소 방법도 컨볼루션 레이어의 파라미터 수를 감소함으로 뉴럴 네트워크의 최적화를 용이하게 한다. 그러나 컨볼루션 레이어의 채널을 감소하는 방법은 뉴럴 네트워크에 입력되는 영상의 세부 정보를 유지한다. 두 번째로는 D_{RP} 와 G 가 최소 극대화 게임 (minimax game) 을 유지하기 위해 D_{RP} 는 충분한 수의 파라미터가 필요하다. D_{RP} 의 파라미터 수가 부족한 경우 I^{HR} 과 I^{SR} 을 구분하지 못하는 문제가 발생한다. 따라서 D_{RP} 는 입력 영상과 가까운 컨볼루션 레이어부터 점차적으로 컨볼루션 레이어의 채널 수가 감소하는 구조를 갖는다. D_{RP} 의 파라미터 수가 매우 많아서 완전 연결 레이어 (fully connected layer; FC layer) 에 많은 수의 특징이 입력되는 경우 FC 레이어의 파라미터 수 또한 많게 되며 이는 D_{RP} 의 과적합화

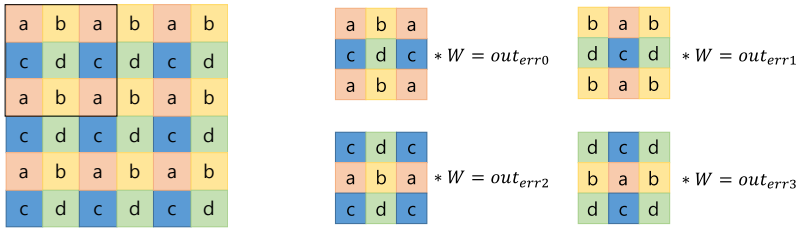


그림 5.2 I^{SR}의 오류 $fmap$ 예시. (a) 네 가지 종류의 입력 오류; (b) 컨볼루션 연산 결과 오류 종류

(overfitting)과 모드 붕괴 (mode collapse) 를 야기한다. 제안하는 D_{RP} 구성 방법은 FC 레이어의 파라미터 수가 과도하게 많아지는 것을 방지한다. 더불어 D_{RP}의 숨겨진 FC 레이어는 [33] 의 방법을 적용하여 모두 제거 한다. D_{RP} 를 적용한 GANs 구조는 최초의 GANs 구조와 동일하다. 따라서 D_{RP}적용시 가치 함수 (value function) $V(D_{RP},G)$ 또한 최초의 GANs의 가치 함수 $V(D,G)$ 의 D 를 D_{RP} 로 치환 한 것과 동일하다.

그림 5.2는 D에 I^{HR}과 I^{SR}의 차이인 오류 $fmap$을 입력하는 예시를 보인다. I^{HR} 및 I^{SR} 은 동일한 컨볼루션 레이어를 통과 하며 각각의 영상이 컨볼루션 레이어 (Conv)를 통과한 결과는 $I^{HR} * Conv$ 와 $I^{SR} * Conv$ 이다. 컨볼루션 레이어를 통과한 $fmap$ 간의 차이는 $(I^{HR} - I^{SR}) * Conv$로 D에 오류 $fmap$을 직접적으로 입력 하는 것과 동일하다. 그림 5.2(a)는 D로 입력되는 오류 $fmap$ 을 의미한다. 이 오류 $fmap$에는 3×3 컨볼루션 필터가 적용 된다. 단순한 설명을 위해 오류 $fmap$은 네 가지 종류의 오류가 존재 한다. 그림 5.2(b)는 그림 5.2(a)의 컨볼루션 결과를 보인다. 스트라이드의 크기가 2인 경우 그림 5.2(b)의 네 종류 오류 중 한 종류만 뒤쪽의 컨볼루션 레이어로 전달 된다. 해당 컨볼루션 레이어 이후의 D 의 모든 컨볼루션 레이어의 스트라이드 크기가 1인

경우에도 선택된 한 종류의 오류 이외의 나머지 종류의 오류는 전달되지 않는다. 네 가지 오류 중 out_{err0}이 뒤 쪽의 컨볼루션 레이어로 전달된 경우 I^{HR}과 I^{SR}은 out_{err0} 만을 기준으로 판별 되며 이는 G가 out_{err0} 만을 제거하도록 학습 된다. 제안하는 D_{RP}를 적용하는 경우 G는 $out_{err\{0,1,2,3\}}$ 모두를 최소화 하는 방향으로 학습 된다.

5.2 해상도 보존 콘텐트 손실

SRGAN은 적대적 손실과 더불어 VGG19 네트워크를 기반으로 한 콘텐트 손실 사용한다. SRGAN은 VGG19 네트워크의 16개의 컨볼루션 레이어 중 마지막 레이어의 MSE를 기반으로 콘텐트 손실 ($l^{SR}_{VGG/16}$) 을 사용 한다. D와 마찬가지로 VGG19 네트워크는 16개의 컨볼루션 레이어 사이에 네 개의 최대 풀링 레이어 (max pooling layer) 들이 있으며 마지막 컨볼루션 레이어에서 출력되는 $fmap$ 내 특징의 개수는 $K_0 \times L_0 \times 2\left(=\frac{512}{2^4 \times 2^4}\right)$ 이다. 이 특징의 개수는 입력 영상의 픽셀 수인 $K_0 \times L_0 \times 3$ 보다 작은 숫자이다. 이는 입력 영상이 VGG19 네트워크를 통과하면서 영상의 세부 정보들이 최대 풀링 레이어에서 손실 됨을 의미한다. 본 장은 VGG19 네트워크의 최대 풀링 레이어로 인해 손실되는 정보들을 방지하기 위한 해상도 보존 콘텐트 손실 ($l^{RP}_{VGG/i}$) 를 제안한다. 제안하는 해상도 보존 콘텐트 손실은 $fmap$의 해상도를 유지하여 영상의 세밀한 정보들을 보존한다. 해상도 보존 콘텐트 손실은 VGG19 네트워크의 최대 풀링 레이어들을 모두 제거함으로 얻을 수 있다. 모든 최대 풀링 레이어들이 제거된 VGG19 네트워크의 i 번째 컨볼루션 레이어의 $ofmap$은 ϕ_i^{RP}로 표현한다. 또한 제안하는 해상도 보존 콘텐트 손실은 기존의 콘텐트 손실의 ϕ_i를 ϕ_i^{RP}으로 치환하여 MSE를 계산한다.

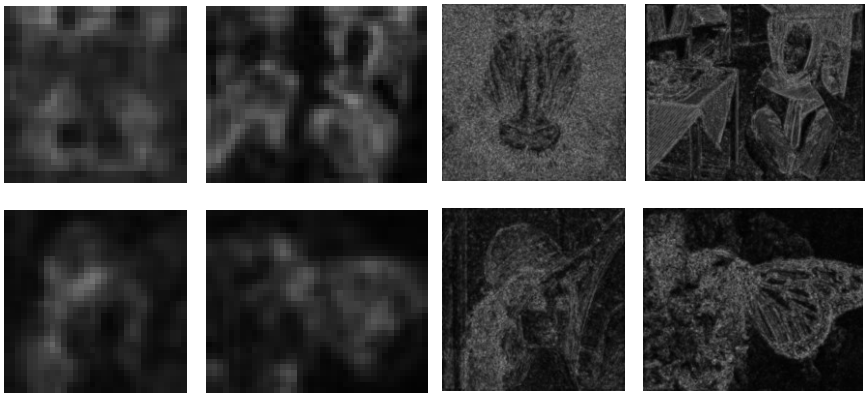


그림 5.3 콘텐트 손실 비교. (a) 기존의 콘텐트 손실 적용시 오류 맵; (b) 해상도 유지 콘텐트 손실 적용시 오류 맵

$l^{RP}_{VGG/i}$는 컨볼루션 레이어와 무관하게 $fmap$의 해상도가 동일하며 뒤쪽의 컨볼루션 레이어를 사용할 수록 더 많은 채널의 $fmap$으로 손실을 계산할 수 있다. SRGAN과 마찬가지로 마지막 컨볼루션 레이어를 사용하여 콘텐트 손실을 계산하는 경우 ϕ^{RP}_{16}은 $K_0 \times L_0 \times 512$개의 특징을 갖는다. 이는 픽셀 기반의 MSE 손실과 비교하여 170.7배 더 많은 정보로 손실을 계산함을 의미한다. $l^{RP}_{VGG/i}$ 도 $l^{SR}_{VGG/i}$과 마찬가지로 미리 학습되어 있는 VGG19 네트워크를 사용하며 제안하는 콘텐트 손실은 G가 I^{HR}과 인지적으로 유사한 I^{SR}을 출력할 수 있도록 학습 한다.

그림 5.3은 기존의 콘텐트 손실과 제안하는 콘텐트 손실을 비교한다. 콘텐트 손실의 자세한 정보를 보이기 위해 I^{HR}과 I^{SR} 간의 차이의 제곱을 한 $fmap$ 결과 이다. 그림 5.3(a)는 콘텐트 손실 오류 맵을 출력하기 위한 영상의 예시를 보인다. 그림 5.3(b)와 그림 5.3(c)는 기존의 콘텐트 손실을 적용한 경우와 제안하는 콘텐트 손실을 적용하는 경우의 오류 맵을 보인다. SRGAN과 동일하게 콘텐트 손실은 VGG19 네트워크의 16번째 컨볼루션 레이어의 $ofmap$을 사용하여 계산

한다. 기존의 콘텐트 손실 ϕ_{16}은 $fmap$의 해상도가 최대 풀링 레이어의 효과로 감소하여 그림 5.3(b)와 같이 영상의 세밀한 정보를 손실한다. 반면 해상도 유지 콘텐트 손실을 적용하는 경우 ϕ_{16}^{RP}는 영상의 세밀한 정보 손실이 없음을 확인할 수 있다.

5.3 실험 결과

5.3.1 데이터 세트 및 평가 방법들

제안하는 해상도 보존 판별 신경망 구조 및 해상도 보존 콘텐트 손실의 효과를 검증하기 위해 테스트 영상은 'Set5', 'Set14', 'BSD100'과 'Urban100' 영상을 사용 한다. SR은 확대 비율 2 및 4에 대해서 실험 한다. 테스트 및 학습을 위한 저해상도 영상 I^{LR}은 I^{HR}에 Bicubic 보간법을 적용하여 영상의 높이와 너비를 확대 비율만큼 축소하여 I^{SR}의 해상도가 I^{HR}의 해상도와 동일하다. 해상도 보존 판별 신경망 구조와 콘텐트 손실은 SISR 이외에도 deblurring 에도 적용 가능하다. 본 장에서는 deblurring 방법 중 DeblurGAN (deblurring generative adversarial network)에 D_{RP}와 해상도 유지 콘텐트 손실을 적용한다. DeblurGAN의 실험은 GoPro 데이터 셋을 사용 한다.

G가 출력하는 I^{SR}과 deblurred 영상 (deblurred image; I^{DB})는 PSNR과 SSIM을 계산하여 객관적인 화질 성능을 측정 한다. SR의 실험 결과는 SRGAN과 동일하게 I^{SR}과 I^{HR}의 주변부 4 픽셀씩 제거하여 Y 채널에 대해 PSNR 및 SSIM을 계산하며, deblurring 실험은 주변부의 픽셀 제거 없이 Y 채널에 대해 PSNR과 SSIM을 계산 한다. SR 실험을 위한 환경은 TensorFlow를 사용 하여 구성 한다. 본 장에서는 TensorFlow를 사용하여 구현한 SRResNet과 SRGAN을 각각

SRResNet-TF와 SRGAN-TF로 표현 한다. DeblurGAN 실험은 저자가 공개한 PyTorch 환경을 사용하여 실험 한다.

5.3.2 해상도 유지 SRGAN의 성능 평가

5.3.2.1 학습 방법

D_{RP} 및 G는 두 개의 NVIDIA GTX Titan X Pascal GPU 들을 사용하여 학습 된다. 두 네트워크를 학습하기 위한 영상은 DIV2K 데이터 셋을 사용 한다. DIV2K 데이터 셋은 800장의 고 해상도 영상으로 구성 된다. 표 5.1은 SRResNet과 SRGAN 네트워크의 PSNR 및 SSIM을 비교한 결과를 보인다. PSNR과 SSIM을 비교하기 위한 영상은 원본 SRGAN [14] 의 테스트 영상과 동일한 'Set5', 'Set14' 와 'BSD100' 영상들을 사용 한다. SRResNet-TF의 평균 PSNR은 [14]와 [34]의 평균 PSNR과 0.02 dB가 차이 남으로 이 차이는 무시할 수 있다. 더불어 SRGAN-TF의 PSNR은 [34]의 SRGAN과 평균 PSNR 차이가 0.04 dB로 이 또한 무시할 수준의 차이 이다. 따라서 본 장부터 SRResNet과 SRGAN 실험은 각각 SRResNet-TF와 SRGAN-TF를

표 5.1 SRResNet 및 SRGAN 네트워크의 PSNR 및 SSIM 비교

Test set	SRResNet			SRGAN	
	[14]	[34]	TF	[14]	TF
	PSNR/ SSIM	PSNR/ SSIM	PSNR/ SSIM	PSNR/ SSIM	PSNR/ SSIM
Set5	32.05/ 0.9019	32.05/ 0.8910	32.00/ 0.8917	29.40/ 0.8472	29.52/ 0.8413
Set14	28.49/ 0.8184	28.53/ 0.7804	28.55/ 0.7793	26.02/ 0.7397	26.32/ 0.7015
BSD100	27.58/ 0.7620	27.57/ 0.7354	27.55/ 0.7340	25.16/ 0.6688	25.06/ 0.6353
Average	27.87/ 0.7745	27.87/ 0.7472	27.85/ 0.7460	25.44/ 0.6846	25.40/ 0.6517

기준으로 한다.

SRResNet과 SRGAN을 학습하기 위한 영상들은 무작위 반전과 회전 (random flips and rotations) 를 통하여 증강 (augmentation) 된다. I^{LR}의 미니 배치 (mini−batch)의 크기는 24×24이며 I^{HR} 의 미니 배치 크기는 I^{LR} 미니 배치의 높이 및 너비에 확대 비율을 곱한 크기이다. 미니 배치 영상들은 RGB 채널을 갖는다. SRResNet의 학습은 초기 학습률을 10^{-4}로 설정하여 10^{6}회 수행된다. 매 2×10^{5}회 학습마다 학습률은 $\frac{1}{10}$으로 감소한다. 이외의 모든 학습 방법은 원본 SRGAN [14]의 방법과 동일하다. I^{LR} 및 I^{HR} 의 범위는 각각 [0,1]과 [−1,1]이다. VGG19 네트워크를 기반으로 한 콘텐트 손실은 0.0006이 곱해져 적용된다. 네트워크들은 Adam 최적화 기법 ($\beta = 0.9$) 을 적용한다. SRResNet 의 학습 이후 판별자 네트워크를 더하여 SRGAN을 학습하며 학습 횟수는 2×10^{5} 회 이다. SRGAN의 초기 학습률은 10^{-4}이며 절반의 학습 후 학습률은 $\frac{1}{10}$으로 감소한다. D_{RP} 및 G 는 1회씩 번갈아가며 학습된다.

5.3.2.2 해상도 유지 판별자 네트워크 구조 선택

D_{RP} 구조는 5.1 장을 따라 구성 되며 컨볼루션 레이어의 필터 개수는 실험적으로 정한다. 표 5.2는 D_{RP}의 구조에 따라 'Set5', 'Set14' 및 'BSD100' 영상들의 평균 PSNR과 SSIM을 보인다. Model 1, 2 및 4는 모두 model 3과 비교하여 낮은 평균 PSNR과 SSIM을 출력 한다. Model 1과 4는 각각 D_{RP}의 파라미터 수가 매우 많거나 부족한 경우를 의미한다. 파라미터 수가 많은 경우 D_{RP} 의 과적화 문제가 발생하여 테스트 영상에 대해서 높은 성능을 내지 못하며 파라미터 수가 적은

경우에는 D_{RP} 의 I^{HR}과 I^{SR}을 구분하는 성능이 낮아서 I^{HR}과 유사한 영상을 출력하도록 G를 학습 하는 것이 불가능 하다. Model 2와 model 3의 차이는 마지막 컨볼루션 레이어의 $ofmap$의 채널 수로 model 2는 model 3과 비교하여 FC 레이어에 2배 많은 특징들을 전달 한다. FC 레이어는 과적화에 약한 단점이 있으므로 model 3이 model 2와 비교하여 높은 PSNR 및 SSIM을 출력 한다. 따라서 D_{RP} 의 구조는 model 3을 사용한다. SR 실험은 확대 비율 2와 4에 대해서 진행 한다. 확대 비율 2에 대해서도 model 3을 변경 없이 사용하여 실험을 진행 한다. 그러나 G 는 두 개의 sub-pixel 컨볼루션 레이어를 내장하고 있으며 하나의 sub-pixel 컨볼루션 레이어는 확대 비율 2에 해당하는 연산을 수행한다. 따라서, 확대 비율 2에 대해서는 SRResNet내 하나의 pixel shuffler layers를 제거 한다. 하나의 pixel shuffler layers는 컨볼루션 레이어와 pixel shuffler 및 활성화 레이어로 구성된다.

표 5.2 해상도 유지 판별자 네트워크 구조에 따른 PSNR 및 SSIM 결과

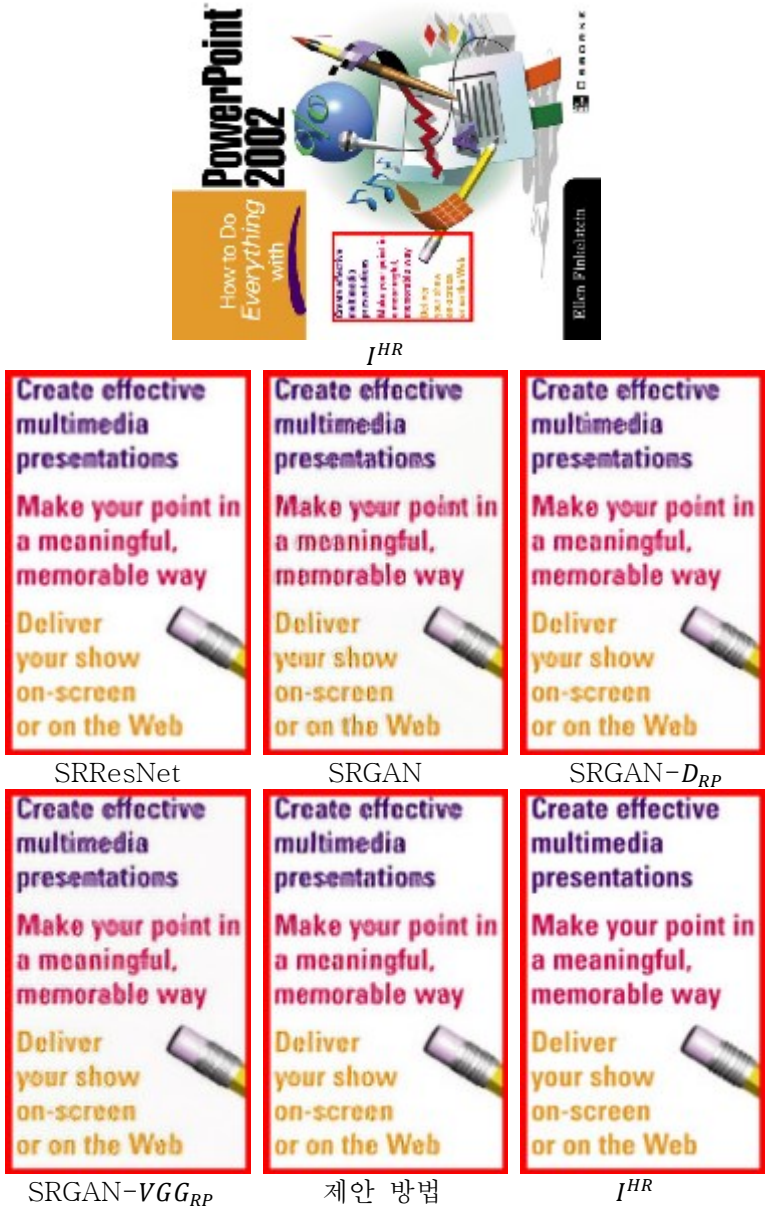
Model	The number of output channels for each convolution layer							
Model 1	256	256	128	128	64	64	32	32
Model 2	256	128	128	64	64	32	32	32
Model 3	256	128	128	64	64	32	32	16
Model 4	64	64	32	32	16	16	8	8

Test set	Model 1	Model 2	Model 3	Model 4
	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
Set5	28.72/0.7919	29.04/0.8400	30.05/0.8490	30.04/0.8470
Set14	25.77/0.6696	25.67/0.6975	26.51/0.7058	26.54/0.7042
BSD100	25.03/0.6444	24.58/0.6348	25.44/0.6468	25.20/0.6380
Average	25.27/0.6536	24.90/0.6508	25.76/0.6622	25.56/0.6546

5.3.2.3 SISR 실험 결과

본 장은 SRResNet, SRGAN, SRGAN−D_{RP}, SRGAN−VGG_{RP} 및 제안하는 방법에 대한 실험 결과를 분석 한다. SRGAN−D_{RP} 는 원본 SRGAN 구조의 D를 D_{RP}구조로 변경한 결과 이며, SRGAN−VGG_{RP} 은 원본 SRGAN 구조에 해상도 유지 콘텐트 손실 ($l^{RP}_{VGG/16}$)을 적용한 결과이다. 더불어 원본 SRGAN 구조에 D_{RP} 및 VGG_{RP}를 적용한 결과를 비교한다. 확대 비율이 4인 경우 I^{LR} 은 텍스트 영역과 같은 영상의 세밀한 정보들을 다수 잃는다. 따라서 실제 영상처럼 보이기 위해서는 과도한 선명화 (sharpening) 이 적용된다. 반면 확대 비율 2의 경우 I^{LR} 에 영상의 세밀한 정보들이 남아있기 때문에 적절한 선명화가 적용되어야 한다.

그림 5.4와 그림 5.5은 확대 비율 2에 대한 실험 결과 영상을 보인다. 그림 5.4의 SRResNet은 I^{HR} 과 비교하여 흐릿한 (blurred) 영상을 출력 한다. SRGAN은 SRResNet와 비교하여 선명한 영상을 출력하나 과선명화의 효과로 시각적 결함 (visual artifact) 가 발생한다. D_{RP} 는 SRGAN의 시각적 결함을 감소하는 결과를 출력 한다. 특히 노란색의 글씨 영역의 시각적 결함이 감소 한다. VGG_{RP}는 보라색 글씨 영역과 빨간색 글씨 영역의 시각적 결함을 감소하는 효과를 보인다. 그리고 제안하는 방법은 D_{RP} 와 VGG_{RP} 의 장점을 결합함으로 시각적 결함을 최소화 하여 I^{HR}과 가장 유사한 영상을 출력 한다. 그림 5.5의 SRResNet은 그림 5.4와 같이 흐릿한 영상을 출력한다. SRGAN은 영상의 왜곡이 발생하여 창틀의 형태가 변하는 시각적 결함을 야기한다. SRGAN−D_{RP} 는 SRGAN의 시각적 결함을 제거하여 SRResNet 보다 선명한 영상을 철력한다. SRGAN− VGG_{RP} 는 SRResNet보다는



I^{HR}

SRResNet | SRGAN | SRGAN−D_{RP}

SRGAN−VGG_{RP} | 제안 방법 | I^{HR}

그림 5.4 확대 비율 2에 대한 'ppt3' (Set 14) 영상의 SR 결과

선명하지만 SRGAN− D_{RP} 보다 흐릿한 영상을 출력한다. 마지막으로 제안하는 방법은 I^{HR}과 가장 유사한 영상을 출력 한다.

그림 5.5 확대 비율 2에 대한 'img001' (Urban 100) 영상의 SR 결과

그림 5.6과 5.7은 확대 비율 4에 대해서 SISR 실험 영상을 보인다. 그림 5.6에서 SRGAN은 물고기의 눈에 시각적 결함을 생성 한다. 반면에 SRGAN-D_{RP}는 이 시가적 결함을 감소한다. SRGAN-VGG_{RP}는 SRGAN이 발생한 시각적 결함은 없으나 SRResNet과 유사하게 흐릿한

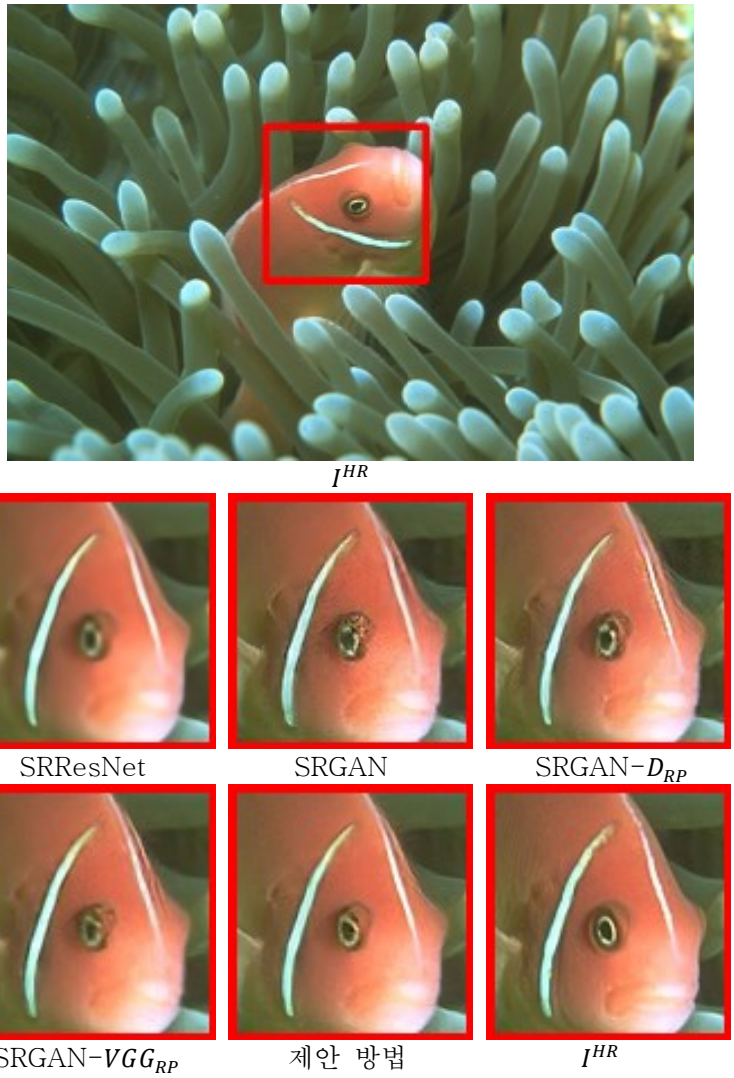


그림 5.6 확대 비율 4에 대한 '210088' (BSD 100) 영상의 SR 결과

영상을 출력한다 제안하는 방법은 SRGAN의 시각적 결함을 제거하며 선명한 영상을 출력하여 I^{HR} 과 유사한 영상을 출력한다. 그림 5.7의 SRResNet은 영상을 흐릿하게 만든다. SRGAN은 과도한 선명화 효과로 인해 영상 내 얼굴에 시각적 결함을 발생한다. SRGAN- D_{RP} 는 SRGAN의 시각적 결함을 완화 한다. SRGAN- VGG_{RP} 는 SRGAN과 비교하여서는 시각적 결함이 적으나 선명도가 SRGAN- D_{RP} 대비 낮은 문제가 있다. 제안하는 방법은 D_{RP} 와 VGG_{RP} 의 장점을 결합하여

그림 5.7 확대 비율 4에 대한 'img007' (Urban100) 영상의 SR 결과

선명하면서도 시각적 결함이 적은 영상을 생성하며 이는 G 가 I^{HR} 과 가장 유사한 영상을 출력하게 한다.

표 5.3은 'Set5', 'Set14', 'BSD100'과 'Urban100' 영상들에 대해서 확대 비율 2 및 4의 평균 PSNR과 SSIM 측정 결과를 보인다. SRGAN- D_{RP} 는 SRGAN과 비교하여 확대 비율 2와 4에 대해 각각 평균 PSNR이 0.44 dB와 0.20 dB 높으며 평균 SSIM은 각각 0.0017과 0.0048이 높다. 또한, SRGAN-VGG_{RP}는 SRGAN과 비교하여 확대 비율 2와 4에 대해 평균 PSNR이 각각 0.59 dB와 0.38 dB가 높으며, 평균

SSIM은 확대 비율 2와 4인 경우에 대해 각각 0.0099와 0.0140이 높다. 제안 방법은 SRGAN에 D_{RP}와 VGG_{RP}를 모두 적용한 방법으로 확대 비율 2와 4에 대해 평균 PSNR이 각각 0.75 dB와 0.32 dB가 높으며 평균 SSIM은 각각 0.0051과 0.0188 높다. 확대 비율 4에 대해 제안 방법과 SRGAN$-VGG_{RP}$의 평균 PSNR을 비교하면 제안 방법이 0.06 dB가 낮으나 이는 무시할 만한 낮은 수치이며 평균 SSIM이 0.0048이 높다. 결과 영상들과 객관적인 성능 수치를 비교한 결과 제안 방법이 I^{HR}과 가장 유사한 영상을 출력 함을 알 수 있다.

5.3.2.3 SISR 결과 영상의 지각적인 성능 평가

SISR의 성능을 평가하기 위해서는 원본과의 픽셀 유사성과 함께 지각적인 (perceptual) 유사성을 사용할 수 있다. 본 논문은 제안하는

표 5.3 확대 비율 2와 4에 대한 평균 PSNR 및 SSIM 측정 결과

Test set	sf	Bicubic	SRCNN	SRResNet	SRGAN	SRGAN $-D_{RP}$	SRGAN $-VGG_{RP}$	제안 방법
		PSNR/ SSIM	PSNR/ SSIM	PSNR/ SSIM	PSNR/ SSIM	PSNR /SSIM	PSNR/ SSIM	PSNR/ SSIM
Set 5	2	33.66/ 0.9299	36.66/ 0.9542	37.72/ 0.9589	35.18/ 0.9354	35.80/ 0.9346	35.71/ 0.9405	36.13/ 0.9400
	4	28.42/ 0.8104	30.48/ 0.8628	32.00/ 0.8917	29.52/ 0.8413	30.05/ 0.8490	30.15/ 0.8546	30.29/ 0.8576
Set 14	2	30.24/ 0.8688	32.42/ 0.9063	33.48/ 0.9160	31.07/ 0.8751	31.45/ 0.8722	31.64/ 0.8835	31.75/ 0.8747
	4	26.00/ 0.7027	27.49/ 0.7503	28.55/ 0.7793	26.32/ 0.7015	26.51/ 0.7054	26.46/ 0.6995	26.52/ 0.7143
BSD 100	2	29.56/ 0.8431	31.36/ 0.8879	32.04/ 0.8975	29.58/ 0.8431	29.91/ 0.8449	30.26/ 0.8569	30.14/ 0.8468
	4	25.96/ 0.6675	26.90/ 0.7101	27.55/ 0.7340	25.06/ 0.6353	25.44/ 0.6468	25.59/ 0.6556	25.49/ 0.6579
Urban100	2	26.88/ 0.8403	29.50/ 0.8946	31.75/ 0.9241	29.24/ 0.8884	29.79/ 0.8906	29.75/ 0.8947	30.19/ 0.8956
	4	23.14/ 0.6577	24.52/ 0.7221	25.98/ 0.7804	24.02/ 0.7130	24.03/ 0.7111	24.28/ 0.7230	24.23/ 0.7289
Average	2	28.47/ 0.8454	30.70/ 0.8936	32.13/ 0.9122	29.65/ 0.8679	30.09/ 0.8696	30.24/ 0.8778	30.40/ 0.8730
	4	24.73/ 0.6685	25.93/ 0.7216	27.00/ 0.7617	24.77/ 0.6797	24.97/ 0.6845	25.15/ 0.6937	25.09/ 0.6985

판별자 네트워크 사용시 I^{SR} 의 지각적 성능 향상 정도를 평가하기 위하여 perception index (PI)를 사용한다 [44]. PI는 PSNR, SSIM과 달리 영상의 지각적인 성능을 측정하는 방법으로 사용 되며 PI가 낮을수록 지각적인 성능이 높음을 의미한다. 표 5.4는 해상도 보존 구조의 지각적인 성능을 측정한 결과를 보인다. 테스트 셋 영상으로는 PIRM (Perceptual image restoration and manipulation) workshop에 사용된 테스트 셋 영상을 사용하며 확대 비율은 4이다. SRGAN−D_{RP}는 SRGAN과 비교하여 PI가 0.028 낮으므로 SRGAN과 비교하여 더욱 원본 영상으로 지각되는 I^{SR} 영상을 출력함으로 해상도 보존 적대적 손실이 기존의 적대적 손실과 비교하여 지각적으로 높은 성능의 영상을 출력하도록 G 를 학습 시킴을 알 수 있다. 반면 해상도 보존 콘텐트 손실은 지각적인 성능보다 G 가 원본 영상과 유사한 픽셀 값을 갖게 함을 의미하며 표 5.4와 같이 RMSE (Root mean square error) 값이 낮으므로 원본과의 유사성을 증가하기 위한 콘텐트 손실의 역할이 개선됨을 확인 할 수 있다.

그림 5.8와 그림 5.9 은 확대 비율 4에 대해서 SRGAN과 SRGAN−D_{RP}와 RPSRGAN의 SISR실험 결과를 보인다. 앞선 5.3.2.2의 실험 결과와 같이 SRGAN은 노이즈로 인한 화질 열화가 존재한다. 특히 그림 5.9의 영상은 영상의 왼쪽 부분에 위치한 건물의 세밀한 부분에 화질 열화가 발생하며 SRGAN− D_{RP} 는 이 화질 열화를 완화하나 해결하지 못한다. 반면 RPSRGAN은 화질 열화를 개선함을 보인다.

표 5.4 해상도 보존 구조의 perception index 실험 결과 (확대 비율 4)

	SRGAN	SRGAN−D_{RP}	SRGAN −VGG_{RP}	RPSRGAN
PI	2.105	2.078	2.265	2.262
RMSE	14.927	14.047	14.279	14.712

SRGAN SRGAN-D_{RP} RPSRGAN

그림 5.8 확대 비율 4에 대한 img061 (Urban100) 영상 SISR 결과

SRGAN SRGAN-D_{RP} RPSRGAN

그림 5.9 확대 비율 4에 대한 001 (PIRM) 영상 SISR 결과

그러나 해상도 보존 콘텐트 손실의 영향으로 인해 원본과의 픽셀 값을 유사하게 하기 위해 영상의 선명도가 감소함을 확인할 수 있다. 그림 5.9도 동일한 경향을 보인다. 건물 옥상의 울타리 부분에 대해 SRGAN과 SRGAN- D_{RP} 는 열화가 존재하는 반면 RPSRGAN은 열화 없이 울타리의 구조를 복원한다. 그러나, RPSRGAN은 영상 중앙 부분의 창문에 blur 현상이 발생한다. 해상도 보존 적대적 손실은 기존의 SRGAN과 비교하여 원본 영상과 유사도를 증가하면서도 선명도를 증가하는 효과를 보이나 노이즈로 인한 화질 열화를 완전히 해결하지

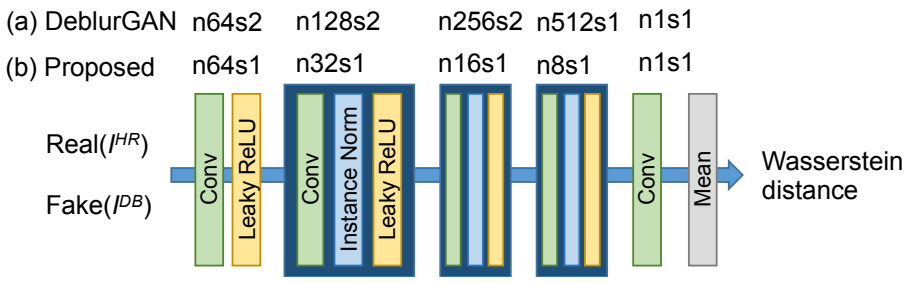


그림 5.10 DeblurGAN의 판별자 네트워크 구조. (a) D의 구조; (b) D_{RP}의 구조

못하므로 해상도 보존 콘텐트 손실을 사용하여 노이즈로 인한 화질 열화를 개선할 수 있다.

5.3.2.4 해상도 유지 구조의 DeblurGAN에의 적용

본 장은 제안하는 D_{RP}와 VGG_{RP}를 DeblurGAN [35]에 적용한 결과를 분석 한다. DeblurGAN은 흐릿한 영상 (blurred image; I^B)를 입력 받아 선명한 영상 (I^{DB})을 출력 한다. 본 장에서는 Kypyn *et al.*이 공개한 $\text{DeblurGAN}_{\text{WILD}}$와 제안하는 방법들을 적용하여 결과를 비교 한다. $\text{DeblurGAN}_{\text{WILD}}$의 실험 환경과 동일하게 영상은 GoPro 데이터 세트의 해상도를 절반으로 감소하여 사용한다. $\text{DeblurGAN} - D_{RP}$는 $\text{DeblurGAN}_{\text{WILD}}$의 D를 D_{RP}로 변경한 결과이며, $\text{DeblurGAN} - VGG_{RP}$는 $\text{DeblurGAN}_{\text{WILD}}$의 콘텐트 손실을 VGG_{RP}로 변경한 결과이다. 더불어 $\text{DeblurGAN}_{\text{WILD}}$에 D_{RP}와 VGG_{RP}를 모두 적용한 제안 방법도 함께 비교한다. 그림 5.10은 DeblurGAN에 적용된 판별자 네트워크의 구조를 보인다. 그림 5.10의 판별자 네트워크의 필터 크기 ($H_i \times W_i$)는 모든 컨볼루션 레이어에 대해 4×4 이다. DeblurGAN의 D는 원본 GANs 구조와 달리 Wasserstein distance를 사용한다 [36]. 또한,

DeblurGAN의 콘텐트 손실은 VGG19 네트워크의 11번째 컨볼루션 레이어를 사용한다. DeblurGAN 실험의 G 구조는 $\mathbf{DeblurGAN_{WILD}}$ 의 G 구조를 그대로 사용한다. D_{RP} 구조의 경우 D_{RP} 와 해상도 유지 콘텐트 손실을 같이 사용하는 경우 D_{RP} 의 평균 풀링 (average pooling) 레이어를 FC 레이어로 변경한다. 판별자 네트워크에 평균 풀링 레이어 사용시 안정적인 학습이 가능하나 수렴하는 속도가 느린 문제가 존재한다 [33]. FC 레이어를 사용하는 경우 판별자 네트워크로부터 생성되는 적대적 손실 (adversarial loss) 는 $\mathbf{10^{-4}}$를 곱하여 적용된다. 표 5.5 는 deblurring 실험 결과를 보인다. 객관적인 성능 비교를 위해 영상의 Y 채널에 대해 PSNR과 SSIM을 계산 한다. $\mathbf{DeblurGAN} - D_{RP}$는 $\mathbf{DeblurGAN_{WILD}}$ 와 비교하여 평균 PSNR은 0.91 dB 높으며 평균 SSIM은 0.0274 높다. $\mathbf{DeblurGAN} - VGG_{RP}$ 는 $\mathbf{DeblurGAN_{WILD}}$ 와 비교하여 평균 PSNR과 SSIM이 각각 1.55 dB와 0.0412 높다. 제안하는 방법은 $\mathbf{DeblurGAN_{WILD}}$ 보다 평균 PSNR이 1.54 dB 높아 제안하는 해상도 유지 구조가 SR이외에 deblurring과 같은 영상 개선에 활용될 수 있음을 보이며, 제안하는 D_{RP} 구조가 원본 GANs 구조 이외에도 Wasserstein GANs에도 적용 가능함을 보인다.

표 5.5 Deblurring에 대한 PSNR 및 SSIM 결과 (GoPro 영상)

	Deblur $\mathbf{GAN}_{WILD}$	DeblurGAN− D_{RP}	DeblurGAN $-VGG_{RP}$	제안 방법
PSNR	27.13	28.04	28.68	28.67
SSIM	0.8504	0.8778	0.8916	0.8971

제 6 장 De-colorization을 적용한 text SISR

본 장은 텍스트 영상에 특화된 SR 학습 및 테스트 방법을 제안한다. 텍스트 영상은 자연적인 영상이 아닌 임의로 합성된 영상 (synthetic image) 으로 영상의 배경 (background) 와 전경 (foreground)인 텍스트의 색상의 조합이 다양하다. 따라서, 다양한 색상의 배경 및 텍스트 조합의 영상들을 학습해야 하며 모든 색상 조합을 CNN에 학습하는 것은 어려운 문제가 있다. 본 장은 기존의 sub-pixel 기반 텍스트 영상 압축 방법 중 de-colorization 을 사용하여 텍스트 영상의 배경과 텍스트의 색상 조합을 각각 흰색과 검은색으로 단일화 한다. 이는 CNN이 적은 개수의 영상으로 학습 되어도 높은 화질의 I^{SR} 을 출력하게 한다.

6.1 Text de-colorization을 적용한 CNN 학습

본 장은 기존의 sub-pixel을 기반으로 한 텍스트 영상 압축 방법 중 효율을 높이기 위한 de-colorization 방법 [18] 을 SISR에 적용하기 위한 방법을 제안 한다. 그림 6.1은 텍스트 영상에 대해 SISR을 수행하기 위해 de-colorization을 적용하는 방법을 보인다. CNN이 흰색 배경과 검은색 텍스트에 대해서만 학습하기 위해서 de-colorization이 적용된 영상 (De-colorized low-resolution image; I_D^{LR})이 CNN에 입력되어야 하며 CNN의 출력 영상 (De-colorized super-resolution image; I_D^{SR})역시 흰색 배경에 검은색 텍스트로 이루어진다. CNN으로부터 출력되는 I_D^{SR}는 de-colorization의 역함수인

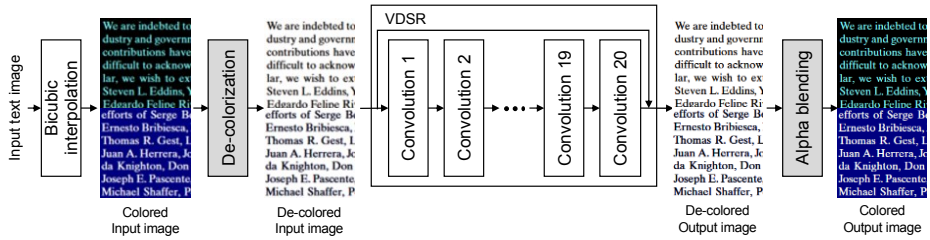


그림 6.1 Text SISR을 위한 de-colorization 적용

alpha blending 과정을 거쳐 de-colorization 적용 이전의 색상이 복원된다. De-colorization과 alpha blending [26] 은 역함수 관계이므로 두 연산은 영상에 대해서 무손실 변환이다.

De-colorization을 적용하는 경우 CNN이 흰색 배경과 검은색 텍스트에 대해서만 SISR을 수행하도록 학습 된다. 따라서 CNN은 training set과 test set의 색상 영향을 받지 않는다. 이는 다양한 CNN을 사용하여 text 영상에 대해 SISR 연산을 하는 경우 흰색 배경과 검은색 텍스트로 이루어진 영상들만 사용하여도 다양한 색상 조합을 갖는 텍스트 영상의 SISR 연산이 가능함을 의미한다.

CNN에 de-colorization을 적용하는 방법은 두 가지가 존재한다. 첫 번째로는 흰 색 배경 및 검은색 폰트로 학습되어 있는 CNN에 de-colorization을 적용이 가능하다. 이 방법은 de-colorization 연산이 다양한 색상 조합의 텍스트 영상 들을 흰색 배경 및 검은색 폰트로 변경하여 CNN으로 전달한다. CNN은 미리 학습된 대로 SISR연산을 수행하며 CNN으로부터 출력되는 I_D^{SR} 는 alpha blending이 적용되어 I^{SR}이 출력 된다. 두 번째 방법으로는 그림 6.1과 같이 de-colorization 연산을 적용한 상태에서 CNN을 학습 시키는 방법이다. 이 방법 또한 CNN은 흰색 배경과 검은색 폰트 영상에 대해 SISR연산을 수행하도록 학습된다.

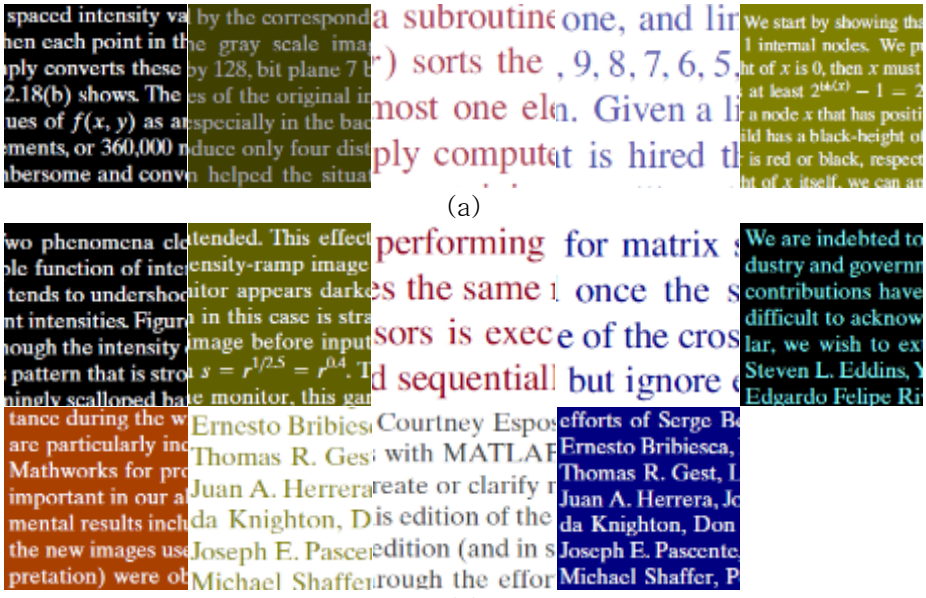


그림 6.2 Text Set 21 training set 및 test set 예시 (a) training set; (b) test set.

6.2 실험 결과

6.2.1 데이터 세트 및 평가 방법들

De-colorization을 적용한 CNN의 성능을 평가하기 위한 영상으로 기존 연구에서 사용한 'ULR-textSISR-2013a' [37] 영상들과 자체 제작한 21장의 영상 (text Set 21) 을 사용한다. ULR-textSISR-2013a 데이터 세트의 test set 영상들은 총 30장의 영상으로 구성 된다. 30장의 영상은 영문 Arial, Courier와 Times New Roman 폰트의 영상으로 구성되며 영상들의 폰트는 10pt와 12pt로 구성된다. 더불어 볼드체와 이텔릭체 영상도 포함한다. ULR-textSISR-2013a 데이터 세트의 학습 영상들은 36장의 영상들으로 구성 되며 테스트용 영상과 같이 영문 Arial, Courier 및 Times new roman 폰트에 대해서 볼드체, 이텔릭체 및 기본 서체를 포함한다. 그러나 ULR-textSISR-2013a

영상들은 모두 흰색 배경에 검은색 텍스트로 이루어져 있는 단순한 색상 조합을 갖는 영상들이다. 따라서 제안하는 de-colorization의 효과를 판단하기 위하여 text Set 21 영상들을 구성 하며 이 영상들은 배경과 텍스트 색상이 다양한 영상들을 웹사이트와 pdf 파일들로부터 취득 한다. Text Set 21의 학습용 영상들은 총 160장의 영상으로 구성되며 테스트용 영상과 같이 웹사이트 및 pdf 파일로부터 영상들을 취득 한다. 그림 6.2는 Text Set 21의 학습용 영상들의 종류와 테스트 세트 영상들의 예시를 보인다. 학습용 영상들은 다섯 가지 종류의 영상으로 구성 된다. 테스트 세트 영상들은 학습용 영상들의 종류를 포함하여 총 아홉 가지 종류의 영상들로 구성 된다.

De-colorization을 적용할 CNN 구조는 VDSR을 사용한다. 원본 VDSR은 Y 채널에 대해서만 연산을 수행한다. 텍스트 SISR을 위해서 VDSR의 입력 채널과 출력 채널의 수를 3개로 하여 RGB 채널을 출력하게 한다. 이 3채널 VDSR을 학습하기 위해서 자체 취득한 160장의 다양한 색상 조합의 training set을 사용한 text용 VDSR의 실험 결과와 함께 ULR-textSISR-2013a의 training set을 사용하여 학습한 text VDSR의 결과 또한 비교한다. 텍스트 SISR의 성능을 비교하기 위해서 PSNR과 SSIM을 사용하며 결과 영상을 비교함으로 제안하는 방법의 효율성을 보인다.

6.2.1 텍스트 SISR 실험 결과

본 장은 텍스트 영상의 SISR 결과를 보여 de-colorization이 텍스트 영상의 SISR에 효과적임을 보인다. SISR의 성능을 측정하기 위한 방법으로 PSNR과 SSIM을 사용하며 SISR 결과 영상을 비교한다. 텍스트 SISR 실험은 ULR-textSISR-2013a 학습 영상들로 네트워크를 학습한 경우와 text Set 21의 학습 영상들로 네트워크를 학습한 경우에 대해 실험 결과를 보인다. VDSR은 영상의 Y 채널에 대해서 super-resolution 연산을 하는 구조이다. VDSR-color 및 VDSR de-color는 VDSR 구조의 network에 RGB 채널을 입력 한다. VDSR-color는 RGB 영상을 전처리 과정 없이 VDSR에 입력하며 출력 영상에 대해서도 후처리 과정이 존재하지 않는다. VDSR de-color는 RGB 영상에 de-colorization 전처리 연산이 적용된 흑색 텍스트와 흰색 배경 영상을 VDSR에 입력하며, SISR 결과 영상에 inverse de-colorization 후처리 연산을 적용하여 색상이 존재하는 텍스트 영상으로 변환하는 과정이 있다. VDSR의 ULR-textSISR-2013a 테스트 영상에 대한 SISR 결과는 기존의 기계학습 방법들과 SRCNN과 비교하여 최소 0.44 dB 높은 성능을 보인다. VDSR 구조를 사용하여 Y 채널과 RGB

표 6.1 ULR-textSISR-2013a training set 학습 PSNR (dB) 및 SSIM

Test sets	[16]		[17]		SRCNN [9]			
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM		
ULR-textSISR-2013a	17.77	0.9340	21.66	0.9350	20.81	0.9320		
Text Set 21	-	-	-	-	-	-		
Test sets	VDSR		VDSR color		Trained VDSR		VDSR de-color	
	PSNR	SSIM	PSNR	PSNR	PSNR	SSIM	PSNR	SSIM
ULR-textSISR-2013a	22.10	0.9412	22.08	0.9412	22.06	0.9431	21.99	0.9418
Text Set 21	22.54	0.8969	19.68	0.8471	26.46	0.9584	27.08	0.9601

채널을 사용하는 경우 0.02 dB의 성능 차이가 존재하며, RGB 채널을 사용하는 경우와 de-colorization을 적용하는 경우 0.09 dB의 차이가 존재한다. Trained VDSR의 경우 VDSR color와 0.02 dB의 차이로 SISR 성능 차이가 미미하다. De-colorization 적용시 흑색 텍스트와 백색 배경 영상으로 학습하고 테스트시 VDSR 기반의 SISR 방법들 중 가장 성능이 낮다. 그러나 Text Set 21 영상들에 대해서 테스트시 de-colorization을 적용한 경우 27.08 dB이며 이는 VDSR의 22.54 dB와 비교하여 4.54 dB가 높다. VDSR은 다양한 색상이 존재하는 텍스트 영상으로 학습되지 않았으므로 text Set 21에 대해서 낮은 SISR 결과를 보인다. 반면 VDSR de-color는 다양한 색상 조합의 텍스트 영상들이 입력 되어도 de-colorization 연산을 통해 흰색 배경과 흑색 텍스트로 변환한다. 따라서 학습되지 않은 색상 조합의 텍스트 영상에 대해서도 화질 저하 최소화가 가능하다. Trained VDSR의 경우도 흰색 배경과 검은색 폰트에 대해서만 학습되어 있으나 de-colorization의 효과로 인해 test Set 21영상들에 대해서 VDSR color 대비 6.78 dB 높다.

그림 6.3과 6.4는 ULR-textSISR-2013a 영상으로 학습한 VDSR 기반의 텍스트 영상 SISR 결과를 보인다. ULR-textSISR-2013a 테스트 영상에 대해서는 VDSR, VDSR color, VDSR de-color 모두 시각적으로 유사한 영상을 출력한다. 이는 학습 영상과 테스트 영상의 색상 조합이 동일하므로 이와 같은 결과가 발생한다. Text Set 21 영상들에 대한 SISR 결과 영상들은 그림 6.4와 같다. VDSR 및 VDSR color는 해당 색상들에 대해 네트워크가 학습되지 않아 영상에 시각적 결함이 발생한다. 반면 VDSR de-color는 테스트 영상들의 색상 조합으로 학습되지 않음에도 불구하고 시각적 결함 없는 결과 영상을 출력한다. 표 6.1의 결과와 그림 6.4의 실험 결과를 토대로 de-

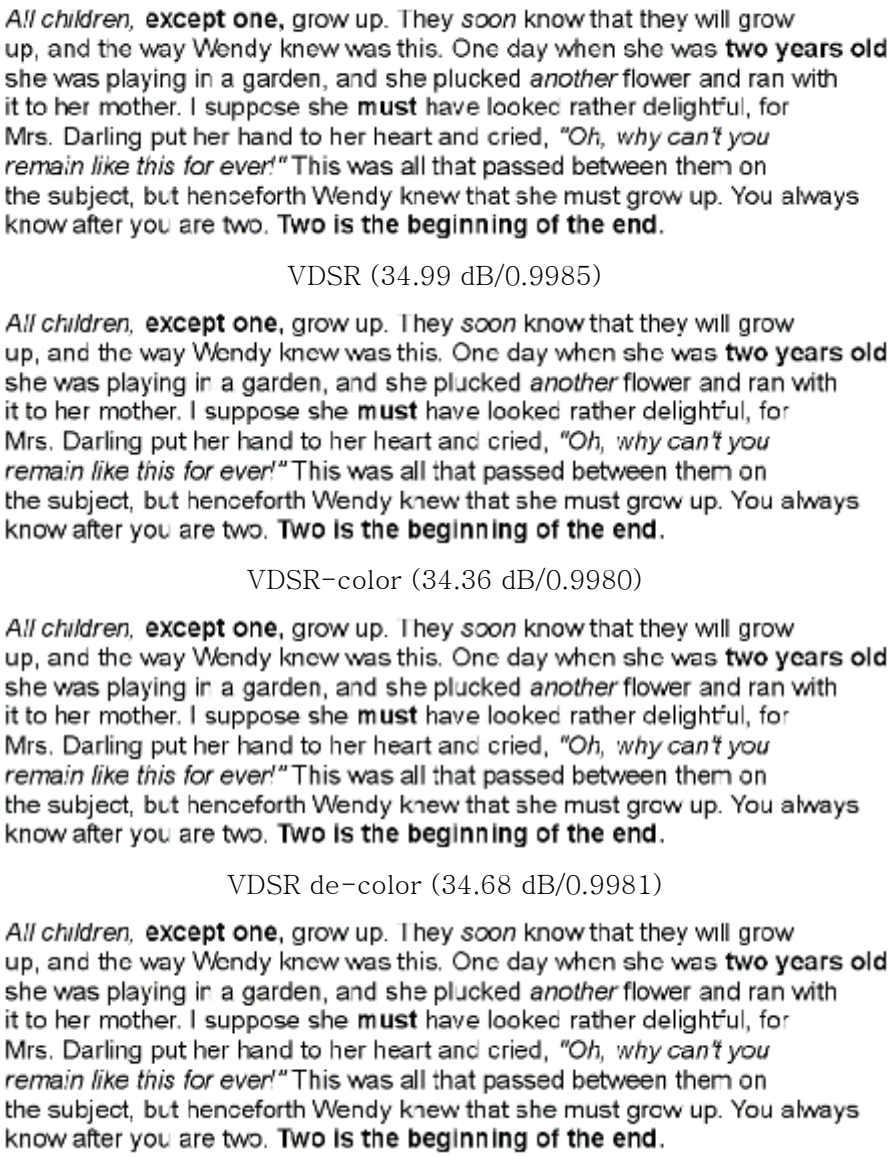


그림 6.3 ULR-textSISR-2013a 학습 및 테스트 결과

colorization 연산은 텍스트 SISR에 적합함을 확인할 수 있으며 기계학습 기반의 텍스트 SISR의 학습용 데이터의 수를 감소할 수

그림 6.4 ULR-textSISR-2013a 학습 및 text Set 21 테스트 결과

있음을 확인 할 수 있다.

표 6.2는 text Set 21의 학습용 영상들로 네트워크를 학습한 경우의 PSNR과 SSIM 결과를 보인다. Text Set 21은 백색 배경에 흑색 텍스트

표 6.2 Text Set 21 training set 학습 PSNR (dB) 및 SSIM

Test sets	VDSR		VDSR-color		VDSR de-color	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
ULR-textSISR-2013a	17.54	0.8691	18.72	0.8912	18.73	0.8932
Text Set 21	22.47	0.9292	28.04	0.9554	35.15	0.9919

색상 조합 영상이 존재하지 않는다. 따라서 ULR-textSISR-2013a 테스트 영상들에 대해서 표 6.1과 비교하여 낮은 PSNR과 SSIM을 출력한다. Text Set 21의 테스트 세트로 SISR 결과 VDSR de-color는 35.15 dB로 VDSR-color와 비교하여 7.11 dB 높은 성능을 보인다. 이는 text Set 21의 학습용 영상이 테스트 셋 영상들의 색상 조합을 모두 포함하지 않기 때문에 VDSR 및 VDSR color 네트워크들은 VDSR de-color와 비교하여 SISR 성능이 낮다. VDSR de-color는 de-colorization 연산을 통해 모든 학습 영상들을 흑색 텍스트와 백색 배경으로 변환하므로 한 종류의 색상 조합에 대해서 정확히 SISR 연산을 수행 하도록 학습 된다.

그림 6.5는 text Set 21 training 영상으로 학습한 후 ULR-textSISR-2013a 영상으로 SISR 연산 수행 결과 영상을 보인다. VDSR은 원본과 비교하여 blur 된 영상을 출력하며, VDSR color는 텍스트 사이에 시각적 결함 픽셀들이 존재한다. 이는 text Set 21의 학습용 영상들이 백색 배경과 흑색 텍스트로 이루어진 영상을 포함하지 않기 때문에 발생한다. 반면, VDSR de-color는 blur 효과나 시각적 결함 없는 I^{SR}을 출력함을 알 수 있다. 그림 6.6는 text Set 21 학습 영상들로 네트워크를 학습한 후 text Set 21 테스트 영상들을 입력하여 SISR 연산 결과를 보인다. Y 채널만 사용하는 VDSR은 원본 영상과 비교하여 어두운 영상을 출력하며, VDSR color는 텍스트 영역에 blur 효과가 존재한다. 반면, VDSR de-color는 원본과 가장 유사한 영상을 출력한다. 앞의 결과들을 토대로 de-colorization 연산을 텍스트 SISR에 적용시 네트학습을 위한 데이터의 수가 부족하여도 높은 SISR

그림 6.5 Text Set 21 학습 및 ULR-textSISR-2013a 테스트 결과

성능을 내도록 학습할 수 있어 텍스트 SISR을 위한 효과적인 방법임을 보인다.

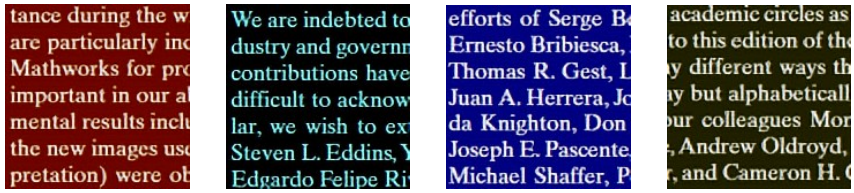


그림 6.6 Text Set 21 학습 및 테스트 결과

제 7 장 결 론

본 연구는 디스플레이 장치에 적용하기 위한 single image super-resolution (SISR) 용 CNN 하드웨어를 위한 최적화 방법들을 제안한다. 디스플레이 장치에 적용되기 위한 하드웨어는 실시간 동작의 제약을 만족하기 위해 외부 메모리 접근 및 하드웨어 리소스 사용량이 제한된다. 또한, 이미 설계되어 있는 SISR용 CNN 하드웨어의 경우 설계 수정이 불가능하다.

본 연구는 외부 메모리 접근이 제한되며 하드웨어 리소스 사용량이 제한된 상황에서의 SISR용 CNN 하드웨어를 설계하기 위한 방법을 제안하며 이를 하드웨어로 구현한 결과를 보인다. 더불어 CNN 하드웨어가 이미 설계되어 있는 상황에서 SISR의 성능을 높이기 위한 방법을 제안한다.

SISR을 위한 CNN 하드웨어 설계는 VDSR을 기반으로 한다. VDSR은 기존의 영상의 큰 컨텍스트 크기 활용과 잔차 신호 학습을 기반으로 기존의 CNN 기반 SISR 방법들 보다 높은 성능을 보인다. 그러나 VDSR은 많은 수의 곱셈기와 내부 SRAM을 사용하여 하드웨어 구현에 어려움이 있다. 본 연구에서는 VDSR의 부분적 수직 방향 컨볼루션 방법을 제안한다. 부분적 수직 방향 컨볼루션은 CNN의 성능 저하를 일으키지 않는 동사에 기존의 듀얼 포트 SRAM을 사용하는 컨볼루션 레이어의 on-chip 메모리를 싱글 포트 SRAM으로 변경할 수 있는 방법을 제안한다. 싱글 포트 SRAM은 듀얼 포트 SRAM과 비교하여 절반의 면적을 사용하므로 on-chip SRAM의 수를 절반으로 줄이는 것과 동일하다.

On-chip 메모리의 크기를 감소하기 위한 추가 방법으로 필터의

형태를 변경하기 위한 방법을 제안한다. 필터의 높이는 on-chip SRAM의 크기와 연관이 있으나 VDSR은 대칭 구조 필터 중 가장 작은 크기인 3×3 크기의 필터를 사용한다. 따라서 컨텍스트 보존 1D 필터 방법과 직사각형 컨텍스트 1D 필터와 부분적 수직 방향 컨볼루션 방법을 적용 한다. 컨텍스트 보존 1D 필터 방법은 SISR 성능을 큰 하락 없이 곱셈기의 수를 2D 필터와 비교하여 55.6%로 감소하고 컨텍스트 보존 1D 필터 적용 이전과 동일한 크기의 SRAM을 사용한다. 직사각형 컨텍스트 1D 필터는 세로 필터의 크기를 감소함으로 SRAM의 개수를 절반으로 하는 동시에 성능 하락을 최소화 한다.

SISR용 CNN의 구조가 고정된 경우 SISR 성능을 높이기 위한 방법은 SRGAN을 기반으로 한다. SISR의 성능을 높이기 위한 기존의 연구들은 CNN의 연산량을 증가 하거나 여러 종류의 서로 다른 손실 함수를 함께 사용한다. 그러나 CNN의 구조를 변경하여 연산 량을 증가하는 경우 하드웨어 재설계가 필요하며 하드웨어 리소스의 제약이 있는 디스플레이 장치에 적용하기에 부적합하다. 더불어 여러 종류의 서로 다른 손실 함수 들을 함께 사용하는 경우는 각각의 손실 함수의 장점들이 결합되는 동시에 단점들도 결합되는 문제가 존재한다. 제안하는 연구는 SISR의 목적인 영상의 주변 정보의 손실을 방지하여 CNN의 구조 변경이나 새로운 손실 함수의 추가 없이 SISR 성능을 개선한다. 영상의 주변 정보 손실을 방지하기 위해 판별자 네트워크의 구조를 해상도 보존 구조로 변경 하였으며 이는 기존의 판별자 네트워크보다 7.91배 작은 구조를 사용한다. 제안하는 판별자 네트워크 적용시 기존 판별자 네트워크를 사용하는 구조와 비교하여 지각적으로 뛰어난 SISR 결과 영상을 출력한다. 또한, 콘텐트 손실을 생성하기 위한 VGG19 네트워크의 구조를 해상도 보존 구조로 변경하여 제안하는

해상도 보존 구조를 갖는 SRGAN은 기존의 SRGAN과 비교하여 영상 4배 확대시 0.32 dB 성능 향상을 보인다.

텍스트 영상을 위한 SISR 방법으로는 기존의 텍스트 영상 압축을 위한 방법인 de-colorization을 차용 한다. CNN의 높은 성능을 위하여는 다양한 색상 조합의 텍스트 영상이 필요하다. 그러나, 모든 색상 조합의 영상을 CNN이 학습 하는 것을 불가능한 문제가 있다. De-colorization은 폰트와 배경의 다양한 색상 조합을 검은 색 폰트와 흰색 배경으로 변환 하는 방법으로 이는 CNN이 한 가지의 색상 조합에 대해서만 학습하고 적용되게 한다. 따라서 학습되지 않은 영상에 대해서도 화질 열화가 없는 SISR 영상 출력을 가능하게 한다.

참고 문헌

[1] D. Glasner, S. Babon and M. Irani, “Super-resolution from a Single Image”, *in Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Kyoto, Japan, Sep. 2009, pp. 349–356.

[2] S. Peled and Y. Yeshurun, “Superresolution in MRI: application to human white matter fiber tract visualization by diffusion tensor imaging”, Magnetic resonance in medicine, vol. 45, no. 1, pp. 29–35, Jan. 2001.

[3] L. Zhang, H. Zhang, H. Shen and P. Li, “A super-resolution reconstruction algorithm for surveillance images”, Signal Processing, vol. 90, no. 3, pp. 848–859, Mar. 2010.

[4] T. Goto, T. Fukuoka, F. Nagashima, S. Hirano and M. Sakurai, “Super-resolution system for 4K-HDTV”, *in Proc. IEEE Int. Conf. Pattern Recogit. (ICPR)*, Stockholm, Sweden, Aug 2014, pp. 4453–4458.

[5] H. Chang, D. Y. Yeung and Y. Xiong, “Super-resolution through neighbor embedding”,in Proc. *IEEE Conf. Comput. Vis. Pattern Recogn. (CVPR)*, Washington, DC, 2004, pp. I-I.

[6] J. Yang, J. Wright, T. S. Huang and Y. Ma, “Image super-resolution via sparse representation”,IEEE Trans. Image Process., vol. 19, no. 11, pp. 2861-2873, May, 2010.

[7] R. Timofte, S. V. De and L. Van Gool, “Anchored neighborhood regression for fast example-based super-resolution”, *in Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Sydney, Australia,

2013,pp. 1920-1927.

[8] R. Timofte, S. V. De and L. Van Gool, “A+: Adjusted anchored neighborhood regression for fast super-resolution”, *in Proc. Asian Conf. Comput. Vis. (ACCV)*, Singapore, 2014, pp 111-126.

[9] C. Dong, C. C. Loy, K. He and X. Tang “Image super-resolution using deep convolutional networks”, *IEEE Trans. Pattern Anal. Mach. Intel.*, vol. 38, no. 2, pp. 295-307, Feb. 2016.

[10] K. Simonyan and A. Zisserman,“Very deep convolutional networks for large-scale image recognition”, *in Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, 2015.

[11] J. Kim, J. K. Lee and K. M. Lee, “Accurate image super-resolution using very deep convolutional networks”, *in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, 2016, pp. 1646-1654.

[12] Y.H. Chen, T. Krishna, J.S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE J. Solid-State Circuits. Vol. 32, No. 1, 2017, pp. 127-138.

[13] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Tenam, "ShiDianNao: shifting vision processing closer to the sensor", *in Proc. Int. Symp. Comput. Arch. (ISCA)*, 2015, pp. 92-104

[14] C. Ledig et. al. “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network”, *in Proc.*

IEEE Conf. Comput. Vis. Pattern Recog.(CVPR), Hololulu, HI, Jul. 2017, pp. 105-114

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets", *in Proc. Adv. Neural Info Process Syst. (NIPS)*, 2014, pp. 2672-2680.

[16] A. Abedi and E. Kabir, "Text image super resolution using within-scale repetition of characters and strokes", Multimed. Tools Appl. Vol. 76, No. 15 , 2017, pp. 16415-16438

[17] A. Abedi and E. Kabir, "Text image super-resolution through anchored neighborhood regression with multiple class-specific dictionaries, Signal, Image and Video Process., vol. 11, no. 2, 2017, pp. 275-282

[18] K. Kim, C.H Lee, and H.J. Lee., "A Sub-pixel gradient compression algorithm for text image display on an smart device", IEEE Trans Consum. Elec. Vol 64, no 12, May, 2018

[19] C. Dong, C. C. Loyand X. Tang, "Accelerating the super-resolution convolutional neural network", *in Proc. Eur. Conf. Comput. Vis. (ECCV)*, Amsterdam, Netherlands, 2016, pp. 391-407.

[20] W. Shi, J. Caballero, F. Huszar, J. Totz,A. P. Aitken and R. Bishop, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network", *in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, 2016, pp. 1874-1883.

[21] M. C. Yang, K. L. Liuand S. Y. Chien, “A Real-Time FHD Learning-Based Super-Resolution System without a Frame Buffer”, IEEE Trans. on Circuits Syst. II: Express Br., vol. 64, no. 12, pp.1407-1411, Dec. 2017.

[22] T. Manabe, Y. Shibataand K. Oguri, “FPGA implementation of a real-time super-resolution system using a convolutional neural network”, *in Proc. IEEE Int. Conf. Field-Program. Technol. (FPT)*, Xi’an, China, 2016, pp. 249-252.

[23] J.-W. Chang and S.-J. Kang, “Optimizing fpga-based convolutional neural networks accelerator for image super-resolution,” *in Proc. ASPDAC*, 2018, pp. 343-348

[24] J.W. Chang, J.W. Kang, and Suk-Ju Kang. "An energy-efficient FPGA-based deconvolutional neural networks accelerator for single image super-resolution." arXiv preprint arXiv:1801.05997(2018).

[25] M. Rastegari, V. Ordonez, J. Redmoon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks", *in Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 525-542

[26] (2013). Reference Software of H.264/AVC Version jm18.4. [Online]. Available: http://iphome.hhi.de/suehring/tml/download/old_jm/

[27] S. Li, J.H. Ahn, J.B. Brockman and N.P. Jouppi, “McPAT 1.0: an integrated power, area and timing modeling framework for multicore architectures”, *in Proc. IEEE/ACM Int. Symp.*

Microarchitecture, 2009, pp. 469–480

[28] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks", *in Proc. Adv. Neural Info. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[29] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up Convolutional Neural Networks with Low Rank Expansions", *in Proc.Br. Mach. Vis. Conf.(BMVC)*, Nottingham, UK, 2014, pp. 88.1–88.13.

[30] S. Schulter, C. Leistner and H. Bischof, "Fast and accurate image upscaling with super–resolution forests", *In Proc. IEEE Conf Computer Vis. Pattern Recognit. (CVPR)*, Boston, MA, 2015, pp. 3791–3799

[31] R, Zeyde, M. Elad and M. Protter,"On single image scale–up using sparse–representations", *in Proc. Int. Conf. Curves Surf.*, Avignon, France, 2010, pp. 711–730.

[32] J. B. Huang, A. Singhand N. Ahuja,"Single image super–resolution from transformed self–exemplars", *in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA 2015, pp. 5197–5206.

[33] A. Radford, L. Metz and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks", arXiv preprint arXiv:1511.06434, 2015.

[34] B. Lim, S. Son, H. Kim, S. Nah and K.M. Lee, "Enhanced deep residual networks for single image super–resolution", *in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*

(CVPRW), 2017, vol. 1, pp. 3.

[35] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin and J. Matas, “DeblurGan: Blind motion deblurring using conditional adversarial networks”, ArXiv e-prints, 2017.

[36] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin and A. Courville, “Improved Training of Wasserstein GANs”, ArXiv e-prints, Mar. 2017

[37] N. Nayef, J. Chazalon, P. Gomez-Kramer, and J.M. Ogier, "Efficient example-based super-resolution of single text images based on selective patch processing", *in Proc. Int. Workshop Doc. Anal. Syst.*, 2011

[38] M. Hailesellasi, and S.R. Hasan, "A fast FPGA-based deep convolutional neural network using pseudo parallel memories", *IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2017, pp. 1-4

[39] J. Qiu *et al.*, "Going deeper with embedded FPGA platform for convolutional neural network", *in Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2016, pp. 26-35

[40] S. Han, H. Mao, and W. J. Dally, “Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding”, *in Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.

[41] Y. Wang, H. Li and X. Li, "Re-architecting the on-chip memory sub-system of machine-learning architecture for embedded devices", *in Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, 2016

[42] S. Vasu, N.T. Madam, and R.A. N, "Analyzing perception-distortion tradeoff using enhanced perceptual super-resolution network", *in Proc. Eur. Conf. Comput. Vis. Workshops (ECCVW)*, 2018

[43] M. S. M. Sajjadi, B. Scholkopf, M. Hirsch, "EnhanceNet: Single image super-resolution through automated texture synthesis", *in Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4501-4510

[44] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli and L. Zelnik-Manor, "2018 PIRM challenge on perceptual image super-resolution", *in Proc. Eur. Conf. Comput. Vis. Workshops (ECCVW)*, 2018

Abstract

On-chip memory reduction in CNN hardware design for image super-resolution

Donghyeon Lee
Electrical and Computer Engineering
The Graduate School
Seoul National University

Unlike convolutional neural network (CNN) for image classification, CNN for single image super-resolution (SISR) receives high-resolution image and generates feature maps which are high-resolution intermediate results. The hardware for accelerating the CNN for SISR is mainly applied to the display device, and the CNN hardware has a streaming architecture in which external memory access is impossible. This causes implementation difficulties due to the limited hardware capacity of the on-chip memory. This paper proposes two methods for designing CNN hardware for SISR using limited hardware resources.

CNN hardware is based on a very deep neural network for super-resolution (VDSR) architecture. By using the partially-vertical order for the convolution layers, simultaneous read and write accesses to SRAM are prevented. The proposed order makes CNN use single-port SRAM instead of dual-port SRAM, and it reduces on-chip

memory area by half. The second method is to change the shape of the filter in VDSR. The size of the on-chip memory is proportional to the height of the convolution filter. However, since the filter of VDSR is the smallest of the symmetric shape, it is impossible to reduce the filter height of the VDSR. To solve this problem, a method of constructing a context-preserving 1D filter and a method of decreasing a vertical filter based on the context are proposed. These proposed methods reduce the size of the SRAM in half.

Two CNN training methods for SISR of natural image and that of text image are proposed. These methods improve SISR performance after the CNN hardware architecture is confirmed. SRGAN (super-resolution generative adversarial networks) is trained by the help of discriminator network to generate realistic natural images. However, SRGAN has the problem of causing visual defects due to over-sharpening. This paper proposes two methods to eliminate the visual defects of SRGAN. First, the resolution-preserving discriminator network structure is proposed. This discriminator network prevents detailed information loss in the network by changing the structure of it. Second, the resolution-preserving content loss is proposed to solve the problem of loss of detailed information of image due to the structure of VGG19 network that causes content loss.

The text image is not a natural image but a synthetic image. The color combination of the font and the background in the image can be variously changed. The existing CNN learning method uses a method of learning various kinds of images to generalize the network.

However, it is impossible to learn all kinds of color combinations on CNN. This paper uses the de-colorization method used in image compression to limit the image to be learned by CNN to a black font and a white background image. As a result, CNN performs SISR operation without visual flaws in the font and background color combination image of the trained image.

Keywords : Single image super-resolution, on-chip memory reduction, real-time hardware, generative adversarial networks, text super-resolution

Student Number : 2012-20828